МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КІРОВОГРАДСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Професор Сидоренко В.В.

Архітектура комп'ютера

Конспект лекцій

Кіровоград

ЗМІСТ

Тема 1

Системний та реальний час комп'ютера IBM PC

Усі сучасні комп'ютери містять годинники (таймери) системного і реального часу.

Системний час утворюється (рис.1) Операційною Системою (ОС) комп'ютери за допомогою інтервального таймера (1), комірок пам'яті (046ch…0470h) (4), годинника реального часу (5)-ВТК, PI CPU (3).

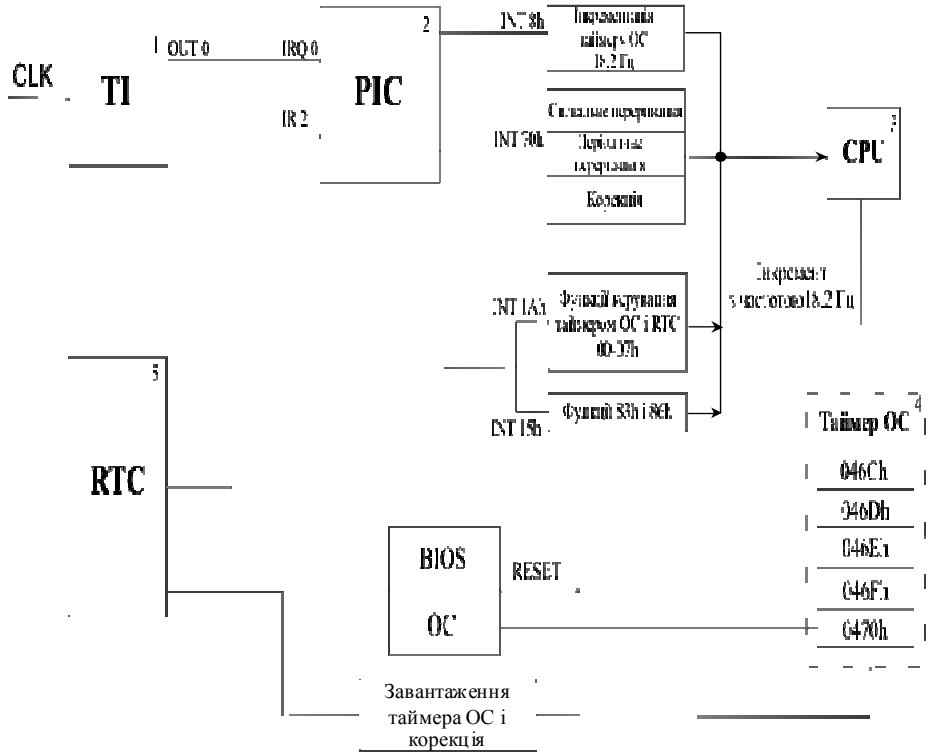


Рис. 1 – Функціональна схема з'єднань пристроїв для утворення системного та реального часу в IBM PC

При увімкненні комп'ютера системний час встановлюється на основі даних годинника реальног часу (5) і надалі працювати незалежно від RTK. При вимкненні ПК дані годинника системи часу втрачаються.

Велика частина прикладних та системних програм використовують цей час для розв'язування системних задач. Наприклад, ОС записує на диск дату та час, створення файлів, встановлює різного роду паузи підчас роботи, тощо.

Прикладні програми можуть періодично виконувати команди, які залежать від часу. Наприклад, зберігати документ на диску через кожні 2 хвилини. Крім того, різні системи керування використовують системний час для своїх потреб (Для визначення термінів різних операцій, планування руху транспорту, тощо). Реальний час комп'ютера утворюється за допомогою мікросхеми (контролера MC 46818, яка живиться від акумулятора, що дозволяє контролювати час незалежно від ввімкнення/вимкнення PC. RTC у своєму складі містить комірки памяті, які дозволяють утворювати календар годинник, будильник, який може видавати сигнал у заданий час, а також зберігати інформацію про склад і налаштування комп'ютера. Функціональна схема з'єднань пристроїв, а також переривань, які утворюють системний та реальний час наведена на рис.1 Розглянемо роботу окремих пристроїв наведеної схеми. Таймер на платі IBM PC використовується для системного часу вихід OUT0 (канал 0), який запрограмований на генерацію прямокутних сигналів з частотою 18,2 ГЦ. Вихід канала 0 з'єднано із входом IR0 PIC. Таким чином генерується апаратне переривання IRQ0(int8h). Таким чином, контролер переривань видає адресу програми обробки цього переривання int8h. Виконуючи цей обробник CPU (3) інкрементує комірки таймера ОС (4):046ch-046fh з частотою 18,2 Гц, а при досягненні кількості тіків, які відповідають добі, (1572480) у комірці пам'яті за адресою 0470h з'являється 1, яка є сигналом для ОС про виклик переривань int1Ah, яке здійснює скид в 0 значень комірок пам'яті 046ch-0470h та корекцію календаря. Таким чином відбувається перехід системного часу на наступну добу. Програма обробки переривання in1Ah здійснює керування таймером Ос, а також встановлення та зчитування даних з RTC.

Для отримання часових затримок та інтервалів використовують функції 83h та 86h програмного переривання int15h. Вихід RTC з'єднано з входом IR2 PIC і апаратне перериванням IRQ2, якому відповідає програмне переривання int70h.

Це програмне переривання GAT адреси программ обробки сигнального переривання (будильник), періодичного переривання, а також корекції.

Розглянемо кожен з пристроїв за допомогою яких реалізовано системний та реальний час в IBM PC.

Інтервальний таймер

0-видача сигналу по закінченню лічби;

1-формування часу запрограмованої довжини;

2-генератор імпульсу певної тривалості;

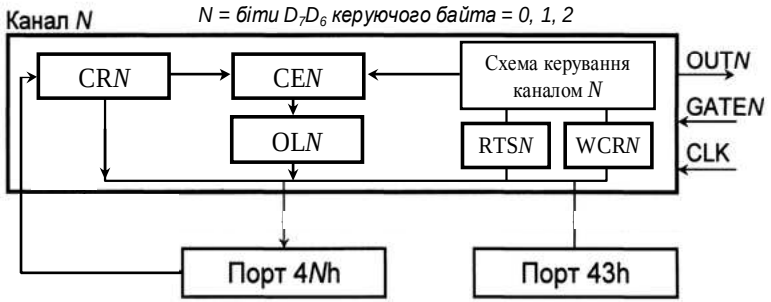


Рис. 2 – Блок-схема канала інтервального таймера

Інтервальний таймер на платі IBM PC, який використовується на системній платі вихід out0,який подається на вхід IR0 PIC (апаратне переривання IRQ0) PIC по входу IR0 видає вектор переривань int8h. По цьому перериванню CPU здійснює інкрементацію комірок пам'яті за допомогою 046ch-046fh. При досягненні заданого числа в трьохбайтних комірках в комірці 0470h встановлюється 1,яка служить сигналом для ОС включення програми коректировки годинника таймера ОС, в тому числі доби і скид в 0 значення комірок пам'яті 046h-0470h.

Початкове значення часу (годинника таймера ОС) при включенні комп'ютера встановлюється за допомогою годинника реального часу RTK. У

відповідь PIC видає програмне переривання in70h, завдяки якому встановлюється початкове значення часу в комірках 046ch-0468h.

Таймер на платі IBM PC містить три незалежні канали. Кожен канал можна представити у вигляді схеми на рис. 2.

Кожен канал таймера містить регістри:

- 16-розрядний регістр лічильника(CE);
- 16-розрядний регістр завантаження константи перерахунку (CR);
- 16-розрядний буферний регістр OL;
- 8-розрядний регістр керуючого слова (WCR);
- 8-розрядний регістр стану каналу (RST);

Канали таймера підключаються до зовнішніх пристроїв:

GATE – керуючий вхід;

CLK – вхід тактової частоти;

OUT – вихід канала.

Регістр CE працює у режимі віднімання і його зміст зменшується по задньому фронту сигнала CLK, при умові, що GATE=1.

Буферний регістр OL – призначений для запам'ятовування змісту регістра CE для зчитування константи перерахунку без зупинки таймера.

Регістр CR – завантажується програмістом і його зміст перезавантажується у регістр CE по сигналу GATE, в залежності від режиму.

Шість режимів роботи таймера з точки зору програмування об'єднуються в ***три групи***:

I група: однократне виконання функцій (режими 0 та 4);

II група: робота з перезапуском (перезавантаженням) (режими1 та 5);

III група: робота з автоперезавантаженням (режими 2 та 3);

У однократному режимі перед початком лічби зміст регістра CR переписується в CE і константа в ch не зберігає початкове значення. Для повторного виконання функції потрібно нове завантаження CR (перепрограмування)

У режимі з пере запуском не потрібне повторне перезавантаження таймеру. По сигналу GATE значення константи з CRпереписується в CE і константа

залишається також у регістрі CR (у режимі автозавантаження значення регістру CR автоматично переписується в регістр CE по закінченню лічби.

У IBM PC використовують всі 3 канали таймера (рис. 3).

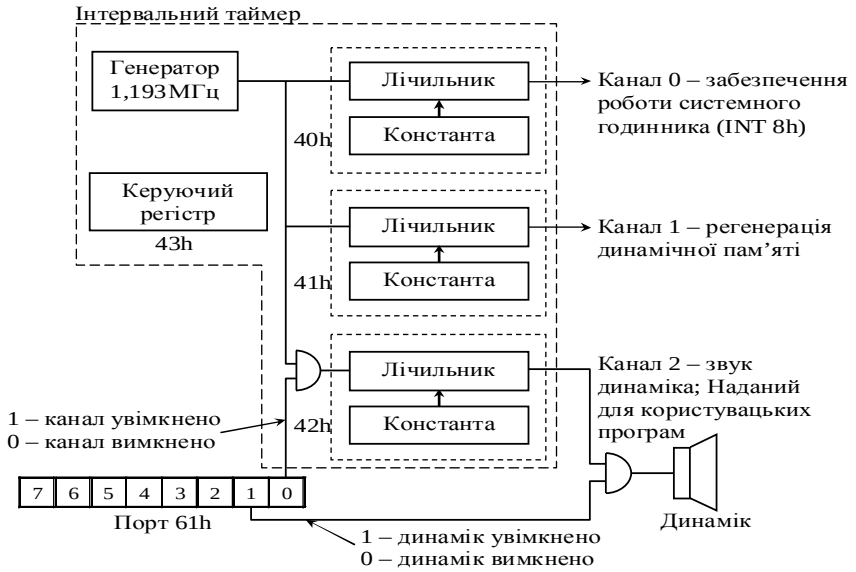


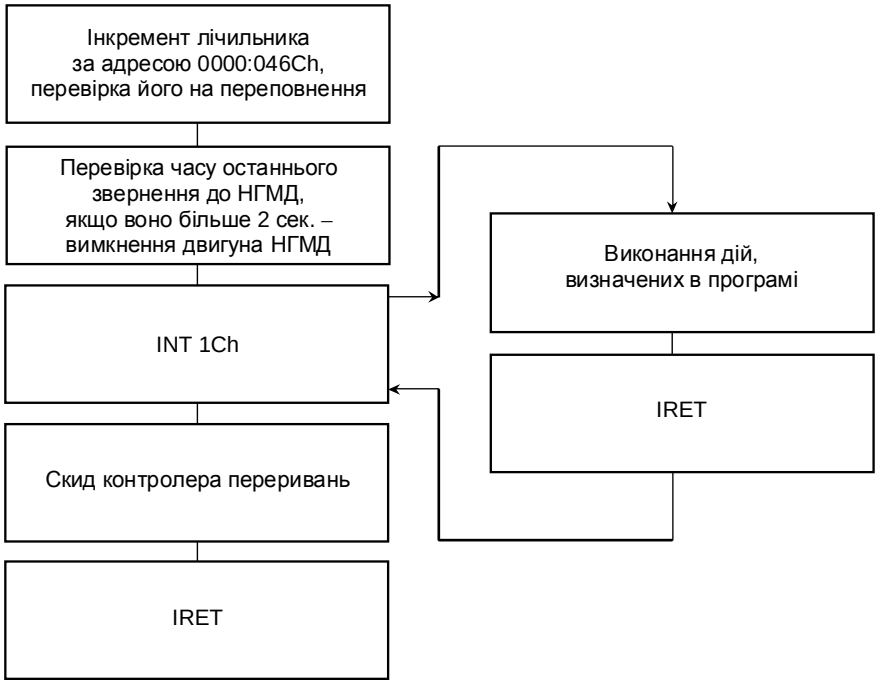
Рис. 3 – Структурна схема призначення каналів таймера на платі IBM PC

Канал 0 – використовують у системних годинниках часу (таймер ОС). Працює у режимі 3 і використовується як генератор імпульсу 18,202 Гц, який визиває апаратну перервуIRQ0 (int8h)

Канал 1 – використовується для регенерації змісту фізичної памяті комп'ютера.

Канал 2 – відданий в розпорядження програміста. Можна відтворювати генерацію різни х звуків, написання фонових програм, або використовувати як генератор випадкових чисел.

Блок-схема переривання int 8h



Програмування інтервального таймера

При включенні комп'ютера вихід інтервального таймера встановлюється в логічну одиницю, а номер режиму, значення і константа перерахунку визначається програмістом.

Програмування таймера слід розділяти на 2а напрямки:

1) Ініціалізація каналу 0, 1 або 2 інтервального таймера – запис керуючого слова і константи перерахунку. Цього достатньо, щоб таймер почав роботу;

2) Читання байту стану каналу і константи перерахунку каналів 0, 1 або 2.

Для програмування таймера використовуються порти 40h, 41h, 42h, 43h. 40h, 41h, 42h – призначені для завантаження константи перерахунку у відповідний канал таймера. Порт 43h – призначений для завантаження керуючого байта, причому номер канула, для якого призначений байт, визначається D6, D7 керуючого байта.

Функціональна схема вибору канала таймера для завантаження керуючого байта і константи перерахунку (рис. 4)

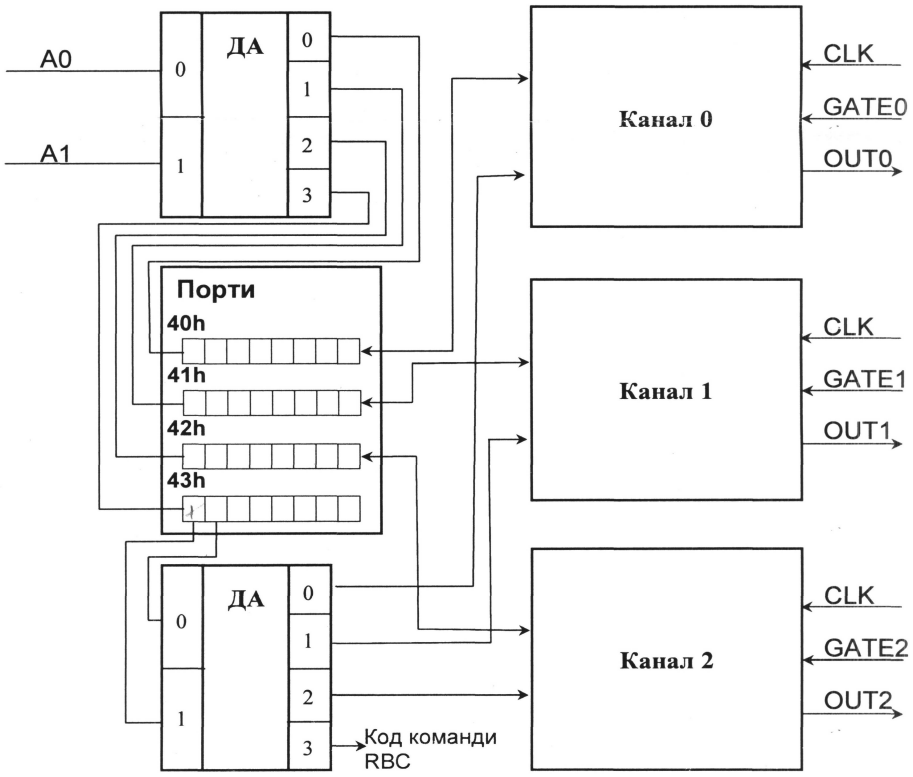
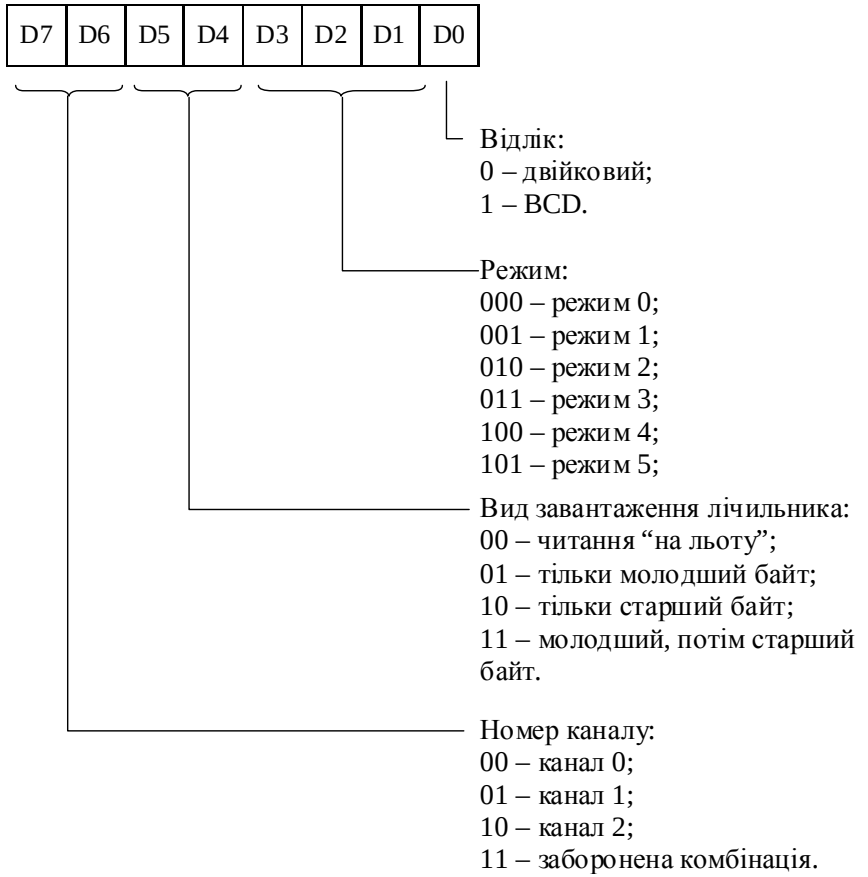


Рис. 4

Комбінація D6=1 і D7=1 в цьому напрямку заборонена.

Формат керуючого байта



Для програмування таймера слід виконувати 2 правила:

1. В кожен лічильний керуючого слова повинно бути записано перед завантаженням лічильника

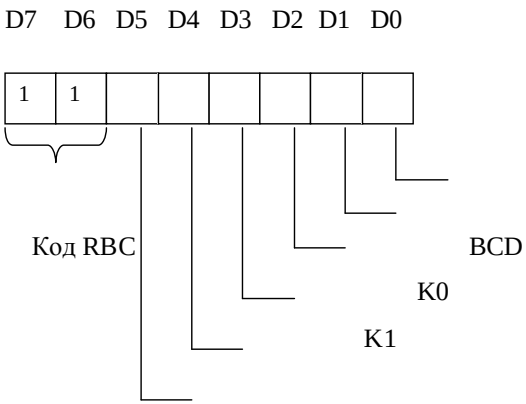
2. Кожен лічильник повинен завантажувати таку кількість байтів і в такому порядку, к вказано в керуючому байті

Розглянемо програмування другого напрямку. Воно полягає в тому, що програмісту необхідно перевірити стан таймера і константу перерахунку таймера. Ця задача програмно виконується двома шляхами:

- зупинкою таймера;
- без зупинки таймера.

Розглянемо 1-й шлях: Читання стану тайме5ра і константи перерахунку відповідних каналів. Для цього необхідно посилати керуючий байт RBC, який утворюється за допомогою бітів D6=1 і D7=1 порту 43h.

Формат команди RBC (Read Back Command)

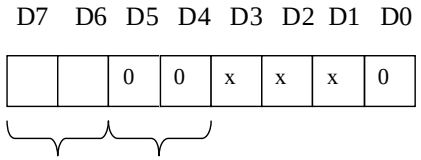


0- читання RST (стан таймера);
1- не читання RST

0-читання константи;
1-не читання

D4 =0
D5 =0 - зоборонена комбінація, код читання «на льоту»

Розглянемо 2-й шлях: Читання константи перерахунку з таймера без його зупинки (читання «на льоту»): для цього програмість в порт 43h посилає команду слідуючого формата:

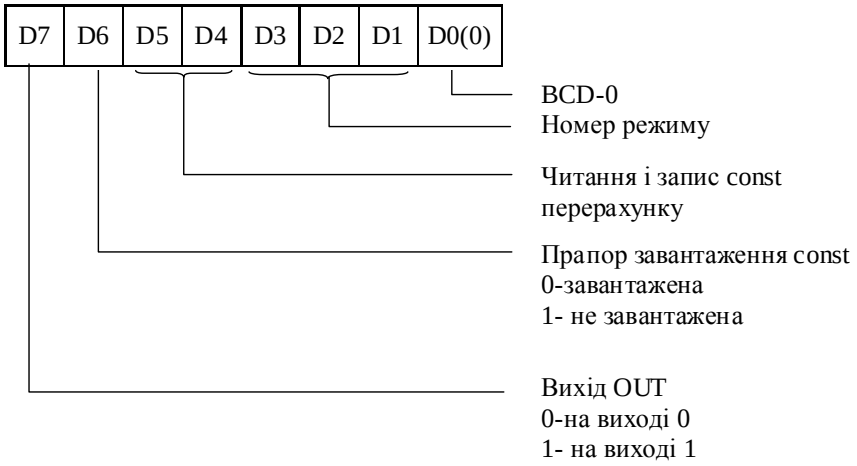


Номер каналу — Код команди читання «на льоту»

00-k0
01-k1
10-k2
11-заборонено

Ця команда дозволяє прочитати зворотнє значення лічильника. Вона не змінює зміст регістру CE. При виконанні цієї команди поточне значення регістру CE фіксується в OL і перебуває там до команди зчитування цього лічильника.

Формат байту стану інтервального таймера



Особливості інтервального таймера:

1. Якщо лічильник програмується на 2 байти читання, то 2 байти повинні бути прочитані;
2. Читання може перериватись записом у той же лічильник. Наприклад:

- читання молодшого байту;
- записати новий молодший байт;
- читання старшого байту;
- записати новий старший байт;

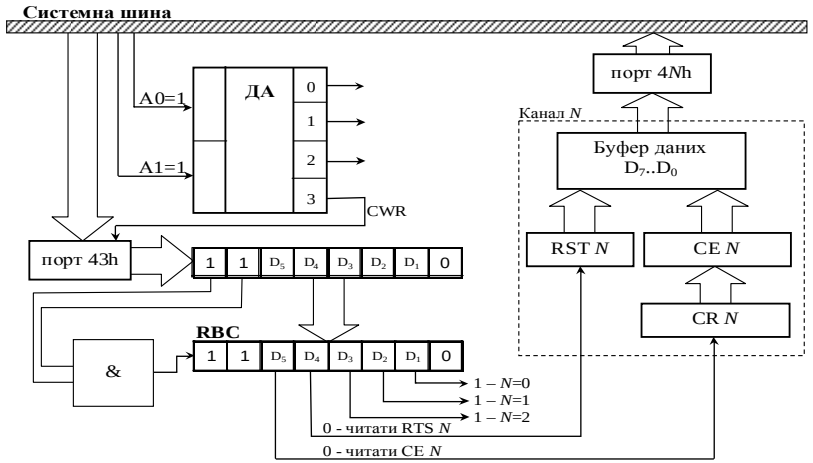
Для читання стану таймера та константи перерахунку в залежності від способу виконання наступної дії:

Спосіб 1: читання RST і константи перерахунку із зупинкою таймера;

Спосіб 2: читання RST і константи перерахунку без зупинки таймера.

Спосіб 1 – Читання RST і константи перерахунку з зупинкою таймера

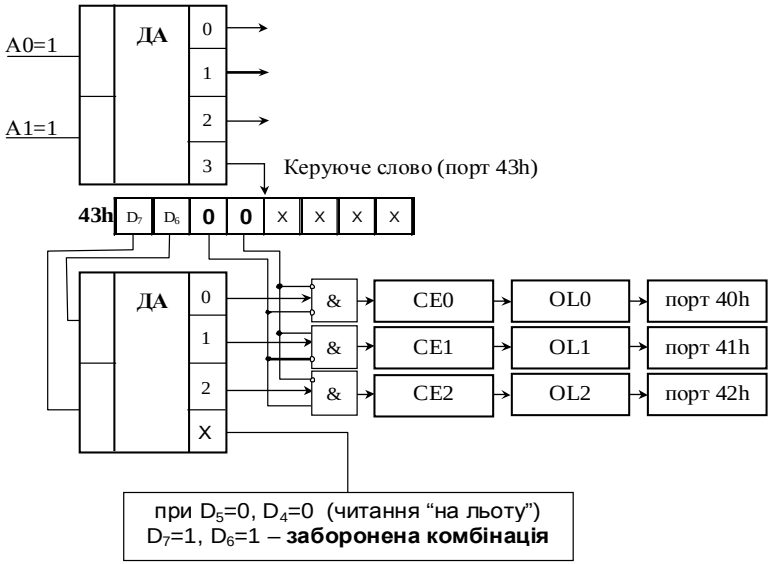
Посилати керуюче слово у порт 43h, в якому D7 і D6=1 і в залежності від того читати регістр стану D4=0. Номер потрібного каналу вказується в бітах D1, D2, D3. Потім необхідно командами читання прочитати із відповідного порту (40h, 41h, 42h) const перерахунку. *Слід звернути увагу*, що якщо в керуючому слові вказано завантаження молодшого і старшого байту, то і читати const слід стільки же разів. D4, D5=0 – заборонено.



Спосіб 2 – Читання константи перерахунку без зупинки таймера

В регістр 43h необхідно записати керуючий байт, в якому D5 і D4=0.

В цьому разі номер канала, в якому слід прочитати const перерахунку вказану в бітах D6, D7 (D6=1, D7=1 – заборонена комбінація).



Наведемо приклад програми, яка перепрограмовує канал 2 інтервального таймера на режим роботи 0 з константою перерахунку 3000.

```
#include <stdio.h>
#include <dos.h>
main (void)
{
 //записуємо керуючий байт відповідно до формату
  outportb(0x43,0xB0);
  outportb(0x40,0xB8);
  outportb(0x40,0x0B);
  printf("Канал 2 перепрограмовано.\n");
}
```

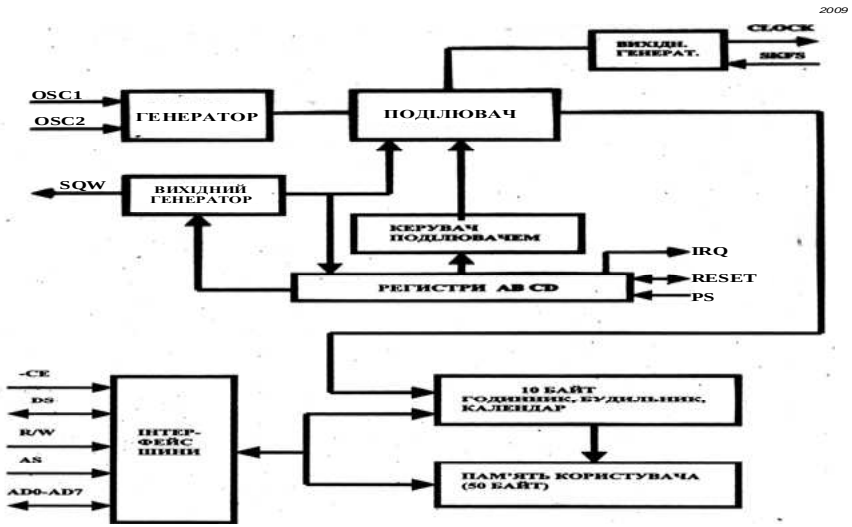
Тема 2

Годинник реального часу (RTC)

Крім інтервального таймера в склад пристроїв, які утворюють реальний та системний час входить також годинник реального часу RTC. Він необхідний для того, щоб зберегти дату і час тоді, коли комп'ютер знаходиться у вимкненому стані. RTC енергонезалежний і живиться від спеціального акумулятора, за станом якого слідкує ОС.

RTC містить календар, а також ряд спеціальних комірок пам'яті (CMOS), які здатні зберігати різні параметри системи при вимкненому комп'ютері. При вмиканні комп'ютера та завантаженні ОС, остання встановлює свій власний годинник (таймер ОС), використовуючи дані з RTC.

Для побудови RTC використовується контролер MC 46818, який використовується в якості RTC і пам'яті конфігурації системи. Контролер містить в собі 64 байти пам'яті CMOS, які складаються із 10 байтів даних годинника і календаря, 4 байти – регістри A, B, C, D (для керування роботою RTC), а також 50 байтів, які використовуються для розміщення інформації про конфігурацію системи.



Таблиця 1 – Адресний простір пам'яті КМОП-мікросхеми

Адреса поля	Кількість байт	Призначення
00h	1	Секунди в BCD
01h	1	Секунди будильника в BCD
02h	1	Хвилини в BCD
03h	1	Хвилини будильника в BCD
04h	1	Години в BCD
05h	1	Години будильника в BCD
06h	1	День тижня (може бути відсутній)
07h	1	Число в BCD
08h	1	Місяць (січень – 1, лютий – 2 і т.д.) в BCD
09h	1	Рік, молодші дві цифри в BCD
0Ah	1	**Регістр A**
0Bh	1	**Регістр B**
0Ch	1	**Регістр C**
0Dh	1	**Регістр D**
0Eh	1	Байт діагностування
0Fh	1	Байт кода скиду процесора
10h	1	Типи HDD (якщо менше 15)
11h	1	Зарезервовано
12h	1	Типи дискет
13h	1	Зарезервовано
14h	1	Склад встановленого обладнання
15h	2	Об'єм базової пам'яті, Кбайт
17h	2	Об'єм розширеної пам'яті, Кбайт
19h, 1Ah	2	Тип першого HDD (якщо більше 15)
1Bh	14	Зарезервовано
2Eh	2	Контрольна сума байтів 10h-2Dh
30h	2	Об'єм розширеної пам'яті, Кбайт
32h	1	Рік, перші дві цифри в BCD
33h	1	Системна інформація
34h	12	Зарезервовано

Доступ до комірок пам'яті CMOS здійснюється за допомогою двох портів – **70h** та **71h**.

В порт 70h спочатку заноситься номер необхідної комірки пам'яті CMOS, до якої додається 80h (D7=1) для заборонення немаскованих переривань NMI. Далі для занесення потрібної інформації в комірку 71h виконується запис або читання потрібної інформації.

RTC в комп'ютері взаємодіє разом з CPU через PIC, формуючи апаратне переривання IRQ8(int70h). RTC має один вихід, який з'єднаний з входом PIC_ф і забезпечує переривання 3х видів:

1. ***Сигнальне переривання (будильник)*** – може бути запрограмоване від 1 разу в секунду до 1 разу за добу;
2. ***Періодичне переривання*** – може біти з періодом 30,517мкс…0,5с (для фонових програм);
3. ***Програмування корегування часу***. Потрібний режим програмується за допомогою регістру B. Кожне з 3з переривань.

Сигнальне переривання

Прграма CPU отримує значення будильника і каледаря шляхом читання байтів через порти 70h і 71h.

В 70h вказуємо адресу необхідного байта +80h(d7=1) і потім з порту 70h читаємо/записуємо потрібну інформацію.

Зміст 10-и байтів 00h-09h – це годинник (календар, будильник) повинен бути двійковим або двійково-десятковим.

При ініціалізації даних комірок памяті треба заборонити корекцію, тобто d7=1 в регістрі B.

Байти годин, календаря і будильника не завжди доступні програмісту. 1 раз в секунду всі 10 байтів перемикаються на корегування, підчас якої неможливо прочитати істинне значення байтів.

Три байти будильника можуть бути використані двома способами:

1. Коли програма встановлює час будильника у відповідних байтах годин, хвилин, секунд, кожен день у вказаний час виробляється переривання будильника;
2. Якщо встановлений розряд дозволяє переривання D5 регістру B;
3. Якщо 2 старших байти встановити в одиницю, то пе6ривання відбуватиметься кожні 2 години.

Періодичне переривання

Дозволяє включити вид IRQ через визначений інтервал від 30 до 517 мкс 30517мкдк до 0,5 сек. Це переривання немає нічого спільного із сигнальним.

Частота періодичного переривання визначається розрядами d0, d1,d2,d3 регістра A.

Періодичне переривання реалізується у всіх системах реального часу, а також використовується для обробки вхідних сигналів кореляційної функції статистичної характиристики і дозволяє працювати у фоновому режимі

Переривання кінця корекції

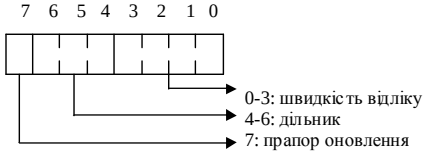
Цикл корекції проводиться кожну секунду, якщо біт d7 регістра B встановлений.

Підчас корекції програма не має доступу до байтів часу, календаря і будильника. Це досягається за рахунок відключення частини пам'яті від системної шини.

RTC має 4 ізольовані комірки пам'яті (порти) (службові регістри A, B, C, D), які призначені для програміста для керування роботою RTC.

Формат регістру A

0Ah – регістр стану RTC:



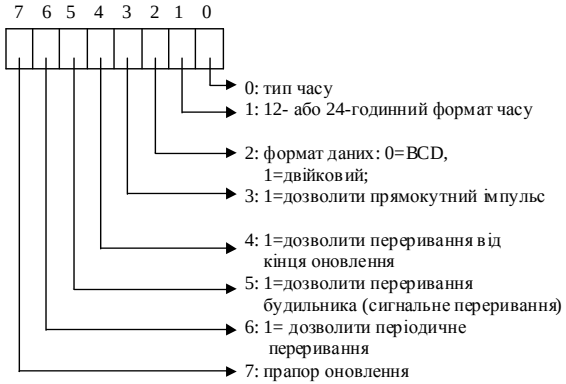
D0, D1, D2, D3 – для вибору першого з 15 виходів поділювана частоти.

D4, D5, D6 – 3 розряди, які дозволяють вибрати необхідну частоту CLK, потрібну для роботи CPU (економія кварцових кристалів)

D7 – розряд контролю поточного циклу корекції. 1- відбувається корекція (дані годинника, календаря, будильника недоступні), 0- дані доступні. D7 доступний тільки для читання: не скидується виводом RESET.

Формат регістру B

0Bh – регістр стану RTC:



D0 – тип часу (0 – зимовий, 1 – літній);

D1 – тривалість доби (1-24 год, 0-12 год);

D2 – формат даних (1 – bin, 0 – 2/10);

D3 – розряд доступу прямокутного сигналу SQR;

D4 – розряд переривання кінця корекції, доступний для читання і запису, дозволяє встановити апаратне переривання IRQ;

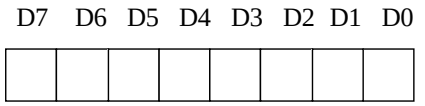
D5 – розряд дозволу сигнального переривання з частотою,заданою в регістрі A, заданої бітами D0, D1, D2, D3;

D6 – розряд дозволу періодичного переривання;

D7 – розряд дозволу корекції;

Регістр B доступний для запису і читання не змінюється RESET.

Формат регістру C



D0, D1, D2, D3 – нулі;

D4 – прапорець кінця корекції;

D5 – прапорець сигнального переривання (1 – співпада сигнал будильника і часу);

D6 – прапорець періодичного переривання

D7 – управління IRQ (1 – IRQ заборонено);

Формат регістру D

Використовується тільки біт D7: 1 – CMOS отримує живлення від акумулятора, 0 – від джерела.

Формат регістру E

0Eh – байт байт результатів початкового тестування.

0Fh – байт стану перезавантаження. Цей байт зчитується після скиду ЦП, щоб визначити, чи не було скиду, викликаного виведенням 80286 з захищеного режима. Він може мати наступні значення:

0 – гарячий рестарт (Ctrl-Alt-Del) або неочікувана зупинка;

1 – зупинка після визначення розміру ОЗП;

2 – зупинка після тестування пам’яті;

3 – зупинка після виявлення помилки паритета пам’яті;

4 – рестарт за запитом початкового завантажувача;
5 – рестарт за скидом контролера переривань та JMP FAR PTR [0:467h];
6, 7, 8 – зупинка після теста захищеного режиму;
9 – рестарт за JMP FAR PTR [0:467h].

Приклад. **Програма переходу на 2000й рік**

Ci

```
#include <conio.h>
#include <dos.h>

int main (void)
{
   char i, rez;
   printf("Перехід на 2000 рік\n");
   i=0x32;
   outportb(0x70,0x80+i);
   outportb(0x71,0x20);
   rez=inportb(0x71);
   if (rez==0x20)
      printf("Перехід виконано успішно!\n");
   else
      printf("Перехід не виконано!\n");
   return 0;
}
```

ASM 86

```
   MOV DX,70h
   MOV AL, B2h      ;(80h+32h)
   OUT DX,AL
   INC DX
   MOV AL,20h
   OUT DX,AL
```


Тема 3

Таймер операційної системи (таймер BIOS)

Для ОС таймер являє собою 32 розрядну комірку пам'яті, яка починається за адресою 046ch-0470h. Ці комірки пам'яті інкрементуються перериванням int8h. При досягненні значення доби цих комірок , ОС скидає таймер комірок в 0 і встановлює прапорець комірки 0470h, яка служить для програмного переривання і корекції таймера.

Для роботи з таймером ОС містить функції 00h-07h переривання int1Ah. Ці функції дозволяють встановити початок значення таймера при включенні комп'ютера , а також дозволяє прочитати поточний зміст таймера,а також зчитати і встановити час і добу RTC.

Функції переривання int1Ah

№ ф-ї	Дія
00h	Зчитування таймера ОС
01h	Встановлення таймера ОС
02h	Зчитування часу RTC
03h	Встановлення часу RTC
04h	Зчитування дати RTC
05h	Встановлення дати RTC
06h	Встановлення сигналізації RTC
07h	Відміна сигналізації RTC

Приклад застосування функції int1Ah

Функція 00h – зчитування таймера ОС на вході Ah=0; ни виході Cx – старше слово лічильника; Dx – молодше слово лічильника; Al=1, якщо з моменту запуску пройшло 24 години. Al(0470h) – прапорець ознаки переповнення.

Mov Ah, 00h

Int 15h

Dx, …

Cx, …

Al, …

Користування функціями Int15h: 01h, 02h, 03h, 04h, 05h, 06h, 07h аналогічно користування функцією 01h.

Таймер Ос дозволяє також утворювати функцію затримки, а також функцію відліку потрібного інтервалу часу.

Для утворення часових інтервалів та функцій затримок часу використовують функції 83h і 86h переривання Int15h.

Розглянемо ці функції: Функція 83h дозволяє запустити таймер Ос на лічбу, вказавши деяку адресу байту в оперативній пам'яті. Програма, яка запустила таймер одразу після запуску, отримає управління. По закінченню заданого часу функція встановлює старший біт заданого байту в 0, що символізує завершення вказівок часового інтервалу. Цю функцію зручно використовувати для організації виконання будь-яких дій, паралельно з відрахування часу. Наприклад обмежити час на введення паролю.

На вході Ah в 83h:

1 – запустити таймер;

0 – зупинити таймер;

Cx – старше слово лічильника (в мікросекундах), Dx – молодше слово; Es:Bx – адреса байту, у якому по закінченні інтервалу часу змінюється біт D7.

Формування затримки: функція 86h

Функція призначена для формування затримок. Дозволяє визначити час затримки в мікросекундах

На вході Ah в 86h:

Cx – старше слово часу роботи лічильника(мкСек);

Dx – молодше слово (мкСек);

Int15h

Засоби MS DOS для роботи з таймером MS DOS використовує 4 функції переривання Int21h із системним таймером:

2ah – визначення поточної дати;

2bh – встановлення дати;

2ch – визначення поточного часу;

2dh – встановлення часу.

Тема 4

Засоби Windows для роботи з таймером ОС

Windows для роботи з пристроями крім переривання використовує функції розширеного програмного інтерфейсу (API).

Розглянемо застосування цих функцій для роботи з таймером ОС.

Для роботи з таймером ОС Windows використовує функції для читання даних:

GetSystemTime

SetSystemTime

Функція GetSystemTime здійснює вибір даних із структури SystemTime , в якій містяться часові дані.

Синтаксис:

Void GetSystemTime (LPSystem.lpst), де LP – long point – вказівник довжини сегмента, який дозволяє вихід програми за межі вказаного сегмента; lpst - long point System Time – вказівник на структуру SystemTime, в яку надходить поточний час.

Вмикаємо файл <widows.h>

Структура SystemTime:

```
Typedef struct_SYSTETIME
{
Word wYear;
Word wMonth;
Word wDayOfWeek;
Word wDay;
Word wHour;
Word wMinute;
Word wSecond;
Word wMillisecond;
}
```


Приклад

```
/*Программа виведення системного часу Ос windows*/

# include<stdio.h>
# include<windows.h>

void main()
{

SYSTIME st;

//читання системного часу Ос Windows GetlocalTime;
//виведення часу у формат год:хв:сек:мілісек

printf (“ Системний час ОС: %d%d:%d”, wHour, wMinute,
wSecond, wMillisecond)

}
```

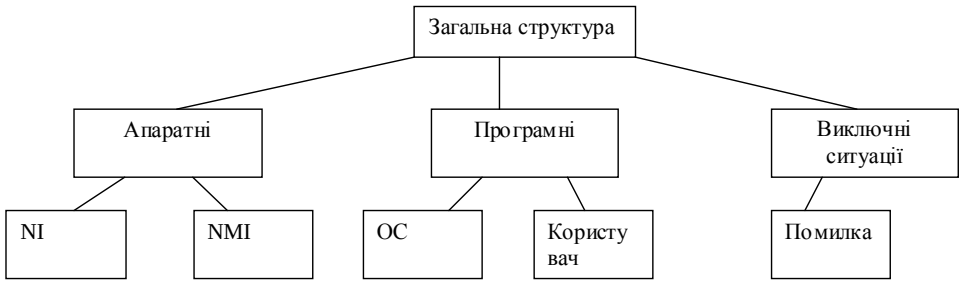
Тема 5

Контролер переривань

З кожним переривання зв'язана та чи інша подія. Система повинна розпізнати, яке переривання за яким номером відбулося і яку відповідну підпрограму треба виконати.

Відомо два види переривань: ***апаратне*** і ***програмне***.

Програмне переривання зручно використовувати для організації доступу для окремих спільних для всіх програм модуля. Прикладні програми можуть самі встановлювати свої обробники переривань для їх послідуючого використовування іншими програмами.



Помилка - являє собою виключні ситуації, які виявляються і обслужуються після вибірки до виконання команди, яка містить помилку.

Пастки – являють собою виключні ситуації, про які повідомляють одразу після виконання команди, яка привела до даної ситуації. CS і IP будуть вказувати на іншу команду, в разі порушення норм виконання команд безпосередньо на команду, яка порушила цей порядок.

Переривання, які визначаються користувачем є прикладом пастки.

Аварійне завершення – це виключна ситуація, що виникає при неможливості точно локалізувати джерело помилки і використати при виявленні помилок.

Апаратне переривання IRQ – використовує масковані переривання MI і немасковані NMI ((переривання інтерфейсів і пам'яті комп'ютера, які завжди проходять через PIC, який має регістр маски, що маскує необхідні переривання.

Програмні переривання – зручно використовувати для організації доступу до окремих і спільних для всіх програмних модулів.

Прикладні програми можуть встановлювати власні обробники переривань. Для цього переривання повинні бути резедентними у пам'яті. Використовуючи переривання з повільними пристроями дозволяю поєднати/О інформації з обробкою даних у ЦП.

Деякі переривання (перші 5) зарезервовані для використання самим процесором для виконання деяких подій. Складання особистих програм обробки переривання і заміна стандартних обробників є відповідальною і складною роботою. Необхідно врахувати усі тонкості роботи апаратури і взаємодії програмного і апаратного забезпечення.

Маскування переривання

Для виконання необхідної послідовності команд при наявності декількох переривань необхідно переривання з вищим пріоритетом замаскувати. Це можна зробити за допомогою команди CLI. Ця команда забороняє тільки масковані переривання, а немасковані завжди оброблюються процесором. Якщо використовується заборона переривань за допомогою CLI, то в кінці обов'язково треба поставити команду STI.

Зміна таблиці векторів переривань

Якщо вашій програмі потрібно змінити обробку деяких переривань, то програма повинна пере назначити вектор переривання на свій обробник. Одним із шляхів вирішення є зміна в таблиці відповідного з векторів переривань.

Послідовність дій для нерезедентних програм обробки переривань:

- прочитати зміст елемента таблиці векторів переривань для вектора з потрібним вам номером;
- запам'ятати цей зміст (адреса старого обробника) у область даних програми;
- встановити нову адресу у таблиці векторів переривань так, щоб вона відповідала початку вашої;
- перед завершенням роботи програми прочитати із області даних адресу старого обробника переривання і записати у таблицю обробника переривання.

Для полегшення роботи по заміні переривання DOS має спеціальні функції для читання переривань і для запису в неї нової адреси. DOS гарантує, що операцію заміни буде виконано вірно.

Апаратне переривання. Для керування схемами пріоритетів необхідно знати внутрішній устрій PIC (I8259). Поступаючи, переривання запам'ятовуються у регістр S запитів IRR.

Найбільш цікавим з програмування є регістр маски IMR і регістр обслуговування ISR.

В машинах класу XT регістр маски переривання 21h; керуючий регістр переривання = 20h.

Для машини AT перший контролер має такі ж адреси,а другий IMR (регістр маски) = A1h. Можна заборонити також немасковане переривання. У комірках AT та Xt передбачені регістри, за допомогою яких програміст може блокувати вхід немаскованого переривання.

Для XT маскування немаскованого переривання керує регістр (порт) за адресою 0A0h. Якщо записати у нього 0, немасковане переривання буде заборонено, а якщо 80h- дозволено.

Для At забороною немаскованого переривання керую бут 7 порта 070h

Типи команд PIC:

Існує 2 типи команд, які програма посилає у PIC:

1) Команда вибору режиму (ICW);
2) Команда управління (OCW), за допомогою яких можна виконати слідуючі операції:

- індивідуальне маскування запитів переривання;
- спеціальне маскування обслуговуючих запитів;
- встановлення статусу рівнів пріоритету обслуговування;
- операції кінця переривання(звичайний,спеціальний і автоматичний);
- читання регістру IMR, ISR, IRR

Команди бувають 3х типів:

1. Маскування запитів переривань;
2. Команди обробки кінця переривання;
3. Опит регістрів і спеціальне маскування;

Байти команди маскування запитів переривання виводяться у ***порти 20h і 21h*** (для першого і другого PIC AT – відповідно).

Команди операцій кінця переривання для 2го і 3го типу (команди обробки кінця переривання, опит регістрів) використовує порти з адресами 20h і A0h, для 1го і 2го контролера відповідно. Для маскування якого-небудь рівня переривання треба записати у регістр масок IMR за адресою 21h або A1h одиницю.

Команди обробки кінця переривання

По команді звичайного кінця переривання програмістом встановлюється у нульовий стан розряд IRS останнього обслуговуваного запиту.

Команда спеціального кінця переривання встановлює у нульовий стан той ISR, номер якого вказано у D0, D1, D2 команди.

Команди циклічного здвигу із звичайним кінцем переривання встановлює у 0 розряд ISR останнього обслуговуваного запиту і цьому ж номеру запиту присвоює нижчий рівень пріоритету.

Аналогічно працює команда циклічного здвигу рівнів пріоритету із спеціальним кінцем переривання. Тільки нижчий рівень пріоритету присвоюється тому входу IRQ, номер якого вказаний у розрядах b0, b1, b2 команди.

Команда циклічного здвигу рівнів пріоритету із спеціальним кінцем переривання встановлює статус рівнів пріоритету без виконання кінця переривання.

Розряди b0, b1, b2 вказують дію пріорітетного кільця.

Команди опиту регістрів і спеціальне маскування

Для отримання змісту IMR необхідно виконати читання портів за адресами 21h і A1h відповідно.

Команда встановлення бітів спеціального маскування блокує дію тих розрядів ISR, котрі замасковані командою маскування індивідуальних пріоритетних рівнів запиту переривання

Спеціальне маскування використовується для такого запиту, який блокується старшим або рівним по рівню пріоритету обслуговуваним запитом, який зберігається в ISR.

Режими роботи PIC:

1. Режим фіксованих пріоритетів (EP) – 0 вищий;
2. Автоматичний зсув пріоритетів;
3. Програмно-керуючий зміст переривання – програміст задає дно переривання;

4. Автоматичне завершення обробки переривання – у звичайному режимі скинути свій біт в ISR. В цьому редимі це здійснюється автоматично і не треба видавати команду завершення EOI. Недолік цього переривання полягає в тому, що в процесі обробки поява повторного переривання втрачається;
5. Режим спеціальної маски – даний режим дозволяє відмінити звичайний режим переривання;
6. Режим опиту – в цьому режимі апаратне переривання не відбувається автоматично. Поява запитів переривання ввизначається читанням IRR.

Програмування PIC

Для програмування Pic використовується 2 порти переривання 20h та 21h. Через ці порти передаються 4 команди ініціалізації: ICW1…ICW4 які визначають режим робоботи PIC і 3 керуючих команди: OCW1…OCW3

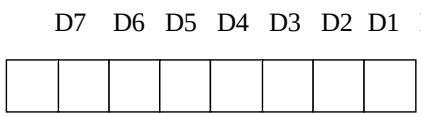
У порт 21h (з парною адресою) вводиться ICW1, ICW2, ICW3(біт D4 показуэ ICW чи OCW).

У порт 21h (з непарною адресою) вводиться ICW1, ICW2, ICW3, OCW1.

Слід пам’ятати: слова ініціалізації повинні слідкувати порядок вводу команд, яких указано для портів 20 та 21 h.

Для 1го PIC ICW3 непотрібно. Наявність ICW4 визначає каскадування

Формат команди ICW1



D0=1 – ICW4 буде

0 – ICW4 відсутнє

D1=1 – 1 PIC

0 – каскад

D2 Адресний інтервал команди call

1 – 4 байти

0 – 8 байт

D3=1 – Запис або скид бітів IRR спеціальною командою

0 – біт в IRR скидається сигналом INTA при установці в ISR

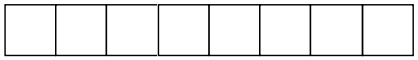
D4=1 – визначає, що вводиться ICW1

0 – визначає, що вводиться ICW2 і ICW3

D5,D6,D7 – розряди молодшого байта адреси A0 =0

Формат команди ICW2

A15 A14 A13 A12 A11 A10 A9 A8



A0=1;

Формат команди ICW3M:

A0=1 - Підключений каскад

0 - Не підключений

ICW3M визначає, до я кого виводу підключена підлегла PIC

Формат команди ICW3S:

D0,D1,D2 – Визначають ідентифікатор підлеглої PIC

Робота PIC без пере ініціалізації

В комп'ютерах IBM PC можливі наступні операції:

- Маскувати/розмаскувати апаратні переривання;
- Змінювати пріоритети рівнів;
- Задавати команду завершення апаратних переривань;

- Встановлювати/скидувати режим спеціальної маски;
- Переводити PIC в режим опиту і зчитувати стан регістру ISR;

Режим опиту PIC

Призначений для використання у випадку, коли кількість зовнішніх пристоїв велика.

В цьому випадку командою OCW3 PIC перводиться в режим опиту і CPU послідовно, за допомогою команди READ опитує стан регістрів ISR або IRR і визначає необхідність робот из пристроєм, який запросив переривання. В режимі опиту сигнал INT і INTA не використовуються.

Тема 6

I/O інформації комп'ютера.
Програмований порт послідовної передачі даних

Дані передаються і приймаються у вигляді послідовних імпульсів з використанням однієї лінії. Для розпізнавання групи символів служить стартовий біт. Потім передаються біти даних у вигляді сигналів високих і низьких рівнів. Останній біт даних може супроводжуватись бітом паритету(бітом парності), який використовується для виявлення помилок. Для виявлення кінця символу включається 1 або більше стопових бітів. Інформаційні біти даних, біт парності, стартовий і стоповий біти – визначаються протоком для обміну інформацією. Приймач і передвавч можуться використовувати один і той же протокол.

Інша важлива характеристика – швидкість передачі. Вона повинна бути однаковою для приймача і передавача. Вимірюється в БОДах.

БОД – це кількість бітів в секунду, враховуючи стартові та стопові біти, а також біт парності.

Основні функції, які виконує UART (УСАПП)

1. Забезпечує перетворення параллельного коду в послідовний при передачі даних і зворотнє перетворення при їх прийомі.
2. Формування стартового, контрольного(паритету) і стопового бітів при передачі даних.
3. Контроль вірності при прийомі стартового , контрольного і стопового бітів.
4. Прийом-передача знаку на фіксованих швидкостях
5. Формування і контроль стану сигналів в інтерфейсі RS232S
6. Організація діагностичної перевірки при використання додаткового обладнання.

Апаратна реалізація COM

Комп'ютери IBM PC мають 2 порти послідовної передачі даних, так як ОС підтримує тільки 2 порти.

Зовнішні пристрої підключені до вводу/виводу через роз'єми DB-9p-і DB-25B

Програмування UART 8250

Адаптер має 10 програмованих однобайтних регістрів, за допомогою яких керується і контролюється обмін інформацією.

Доступ до цих 10 регістрів здійснюється за допомогою 7 адрес: це порти: 3F8h , 3F9h (COM1), 2F8h , 2F9h (COM2).

Асинхронний адаптер вироблює переривання COM1 IRQ (INT0Ch) та COM2 IRQ3 (INT0Bh)

На етапі ініціалізації системи модуль POST BIOS тетсує синхронний адаптер; Їх базові адреси розміщені за адресою 0000:0400h Розглянемо алгоритм і формати портів

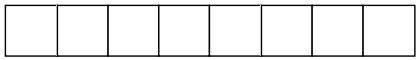
Порт 3F8h

Цей порт відповідає регістру даних, які передаються. Для передачі в порт 3F8h необхідно записати байт, який передається. При прийомі дани від зовнішнього пристрою дані можуть бути прочитані з цього порту. Таким чином, порт 3F8h використовується для прийому і передачі даних.

Прийом і передача даних здійснюється при умові, що у регістрі керування 3FBh біт D7=0. Якщо D7=1, то порт 3F8h використовується для вводу значення молодшого байту дільника частоти тактового генератора.

Порт 3FBh- керуючий регістр

D7 D6 D5 D4 D3 D2 D1 D0



D0,D1 – довжина слова.

D2 – кількість стопових бітів (0 – 1; 1 – 2;)

D3, D4 контроль на парність (00 – не використовується

01- контроль на непарність

11 – контроль на парність)

D5 – фіксація парності 0 –контроль на парність

1 – контроль на непарність

D6 – встановлення переривання

D7 – якщо 1 – порти 3F8h – 3F9h використовуються для завантаження константи поділювача частоти

0 – порти використовуються звичайні порти 3F8h – прийом-пердача, 3F9h - встановлення дозволу обробки переривання.

Регістр керування перериваннями адаптера порт 3F9h

Регістр дозволу переривання використовується для керування перериваннями від асинхронного адаптера або для введення значення старшого байту дільника частоти тактового генератора

Якщо програміст не використовує переривання, то все одно необхідно провести запис в регістр

Регістр керування перериваннями має формат:

Регістр ідентифікації переривань порт 3FAh

По його змісту програміст зможе визначити причину переривань. Формат :

D0 = 1 - немає переривань, що очікують обслуговування

D1, D2 =00 – переривання по лінії стану приймача(помилка виникає при переповненні буфера приймача, помилка парності або формату даних). Скидається після читання даних 3F9h

=01 – дані прийняті і доступні до читання. Скидається після читання порту стану лінії 3FDh

=10 – буфер передавача пустий, скидається після запису даних у порт 3F8h

=11 – стан модему. Встановлюється при зміні стану вхідних сигналів лінії CTS,DSR,R

D3-D7 - не використовуються

Порт 3FCh – регістр керування модемом

Керує станом вихідних ліній DTR,RTS лінією модему OUT1,OUT2, запускає діагностики. Формат:

D0 – лінія DTR

D1 – лінія RTS

D2 – OUT1 : 0 – одиниця на виході

D3 – OUT2 : 1 – заборона виходу переривань

0 – дозвіл виходу переривань

D4 – запуск діагностики(вхід асинхронного адаптера замкнутий на вихід)

D5- D7 – не використовуються

Порт 3Fdh – регістр стану ліній

Призначений для контролю модему при прийомі і передачі інформації. Формат регістру:

D0 = 0- буфер прийому регістру пустий

1- завантажений

D1 – сигнал переповнення при прийомі

0 – переповнення відсутнє

1 – переповнення

D2 – помилка паритету

D3 – стопові біти

0- немає

1- є

D4 – переривання обриву ліній

0- немає

1- є

D5 – буфер передавача

0-заватажений

1 – пустий

D6 – регістр зсуву

0- завантажений

1- пустий

D7 – не використовується

Алгоритм програмування COM-порту

1 – в порт 3FBh послати 80h

2 – в порт 3F8h послати молодший байт, а в 3F9h старший байт дільника частоти

3 – в порт 3FBh послати константу потрібного режиму D7=0

4 – дозвіл всіх переривань в порт 3F9h посилаємо 0Fh

5 – в порт управління модемом 3FCh посилаємо 0

Пункти 1-5 є загальними для прийому – передачі

Підтримка синхронного адаптера в BIOS

Існує 4 процедури послідовної передачі даних, загальних для всіх моделей IBM PC. Ці процедури доступні через переривання INT 14h. Вони мають номера від 0 до 3 і вибираються за допомогою регістрів Ah

№ ф-ї	Значення Ah	Функція
0	Ah=0	Ініціалізація параметрів послідовного порту
1	Ah=1	Передати один символ
2	Ah=2	Отримати один символ
3	Ah=3	Отримати стан послідовного порту та модему

Розглянемо процедури:

Процедура 0:

Встановлює 4 параметри послідовного порту; швидкість передачі даних, парність, кількість стопових бітів, довжину4 сримвола.

Ці параметриобєднуються у 8-бітовий код, який поміщається у регістр Al. Формат Al (На вході AH=0, DX=0- COM1,DX=1-COM2)

Процедура 1:

Передати 1 символ(байт). Ця процедура передає зовнішньому пристрою через послідовний порт 1 символ. Символ розміщується у регістрі Al. Для передачі байта на вході:

AH= 01h DX (номер порту) 0 – COM1,1 – COM2

Al- байт, який передається .

На виході, після виконання Al – зберігається, AH – стан порту асинхронного адаптера. Якзо біт D7 регістру AH

встановлений в одиницю, то виникла помилка.

Процедура 2:

Повідомлення про помилку є відсиланням від норми.

Так як біт D7 вказує, що була помилка, то немає можливості розпізнати TIMEOUT. Для цього слід використовувати повідомлення про стан процедури 2.

На малюнку показана блок-схема DMA. Кожен з каналів К0-К3 має регістри:

16-бітний регістр адреси,14-бітний лічильник і 2-бітний регістр режиму канала. Останні об'єднуються в 1 16-бітний регістр, який зветься лічильником.

До виконання операції КПДП треба ініціалізувати. У регістрові адреси завантажується початкова адреса області пам'яті . В лічильник завантижити кількість передаваємих байт. У регістр режиму канала потрібний режим його роботи (00 – перевірка; 01 – запис; 10 – зчитування;11- заборонена комбінація).

Обмін інформацією може здійснюватись в одному з 4-х режимів:

1) Режим одиночної передачі (Single Transfer Mode) – КПДП звільняє шину і починає перевірку сигналів запитів DRQ і при наявності запитів ініціалізує наступний цикл передачі.

2) Режим блочної передачі (Block Transfer Mode) – Наявність сигналу потрібна тільки до моменту видачі контролером сигнала Dack . Після цього шина звільняється після передачі всього блоку.

3) Режим передачі по вимозі – передача здійснюється дока активний сигнал запиту DRQ, стан якого перевіряється після кожного циклу передачі (з повільними пристроями).

4) Каскадний режим. У цьому режимі 1-ці із виходів канала головного контролера використовуються для каскадування з підлеглим контролером.

Призначення основних каналів

1) I/OR – зчитування I/O;

2) I/OW – запис I/O;

3) A3..A0 – 4 двунаправлені лінії адреси.

Якщо I/OR = 0, байт шини даних завантажується в регістр КПДП по адресі A3..A0.

Якщо I/OW = 0 – зміст адресує мого регістра КПДП передається на шину даних.

I/OR-MEMW – ця пара генерує сигнал для передачі в основну пам'ять .

I/OR-MEMR – ця пара генерує сигнал для передачі із основної зовнішньої пам'яті.

Програмна модель КПДП містить 18 8-бітних регістрів.

Ці регістри розподіляються на регістри, які є загальними для цих каналів і на регістри для кожного каналу окремо.

1) Регістр початкової адреси (Base Adress Register) – в цьому регістрі задається початок адрес ОЗП, з якого починається передача. Регістри містить 16 розрядів і визначає адресу сторінки. І зміщення всередині сторінки.

2) Регістр початку лічильника циклів – в цьому регістрі задається початкове число циклів для програмованого канала.

3) Регістр поточної адреси. Початкове значення заноситься в цей регістр одночасно з регістром початку адреси. Далі треба передати значення поточного регістра за кожен цикл.

4) Регістр поточного значення лічильника циклів. Регістр містить поточне число циклів передачі, які залишилися.

5) Регістр режиму (Mode Register)

D7 D6 D5 D4 D3 D2 D1 D0

D0- D1 – 00 –перевірка

11 – запис у пам'ять

10 – читання за пам’яті

11 – заборонена комбінація

D2 – 0 – автоініціалізація заборонена

1 – автоініціалізація дозволена

D3 – Зміна потомної адреси при обміну

0 – збільшення (інкрементація)

1 – зменшення (декрементація)

D4- D5 – Тип передачі

00 – режим передачі по вимозі

01 – режим одиночної передачі

10 – режим блочної передачі

11 – каскадний режим

D6- D7 – не використовуються

Кожен із 4-х каналів КПДП має свій набір регістрів, які описані вище.

Крім того DMA містить загальний регістр для всіх каналів

Регістр керування:

Формат:

D7 D6 D5 D4 D3 D2 D1 D0

D0 – 0 - заборонити пам'ять в пам'ять

1 - дозволити

D1 - 0 – заборонити фіксацію адреси каналів

1 – дозволити

D2 - 0 – розблокувати DMA

1 – заблокувати DMA

D3 - 0 – нормальна передача

1 – стиск передачі (стиск часу передачі)

D4 - 0 – режим фіксованих пріоритетів

1 – режим циклічного зсуву пріоритетів

D5 - 0 – затримка при запису

1 – режим розширеного запису(якщо біт D3 =1, то D5 ігноруємо)

D6 - 0 – активним є високий рівень DRQ

1 – низький

D6 - 0 – активним є високий рівень сигнала Dack

1 – низький

Регістр стану

Відображає поточний стан запитів і передач по всім 4-ом каналам

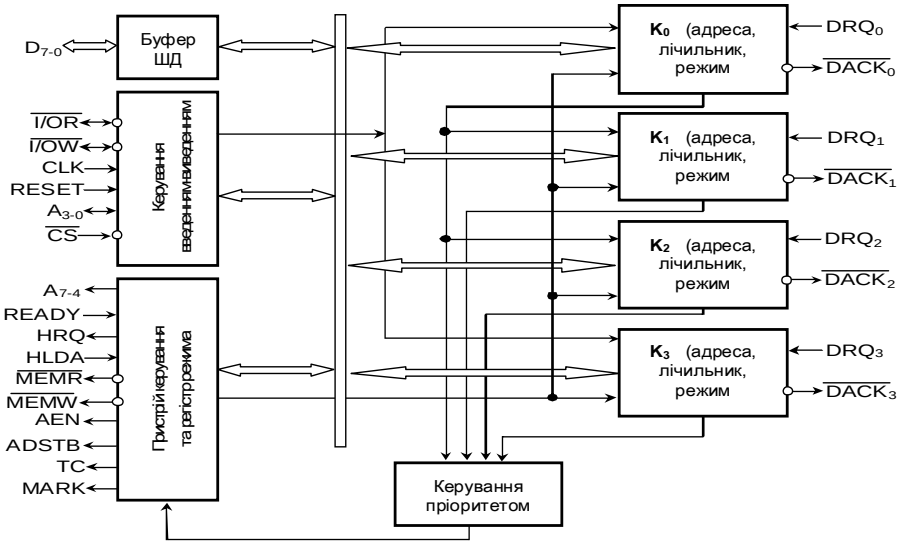
Біти D0- D3 -1-ця вказує канал, який працює.

Регістр маски

4-бітний регістр. Маскує/демаскує свій канал контролера – 1-маскує канал, 0 – дозволяє прийом запита по цьому каналу.

Регістр запитів (Request Register)

КПДП (DMA) в комп'ютерах IBM PC



CLK – вхід для підключення тактового генератора;

CS – вибір кристала;

RESET – скидання мікросхеми;

READY - сигнал готовності;

HLDA - підтвердження захоплення шини;

DREQ3 - DREQ0 - входи запитів на ПДП від зовнішніх пристроїв;

DB7 - DB0 - двунаправленная шина даних з буфером;

IOR - сигнал читання;

IOW - сигнал запису;

EOP - закінчення процесу;

A3 - A0 - адресні входи / виходи;

A7 - A4 - адресні виходи, на які в режимі ПДП передаються відповідні розряди адреси ОЗУ;

HRQ - вихід запиту захоплення шин;

DACK3 - DACK0 - підтвердження ПДП;

AEN - дозвіл адреси.

ADSTB - строб (імпульс) адреси.

MEMR - читання з пам'яті.

MEMW - запис в пам'ять.

Ucc - шина живлення (+5 В).

GND - загальний.

Регістри каналів DMA для IBM PC XT

Кожен канал містить 16-розрядні регістри:

– Регістр поточного значення адреси CAR (Count Adress Key) - містить поточну адресу комірки пам’яті, при використання операцій з обміну даними з використанням DMA;

– Регістр циклів CWR (Count Word Register) - містить число слів – 1 призначених для передачі;

– Регістр зберігання базової адреси BAR (Base Adress Register) - використовується для зберігання базової(початкової) адреси пам’яті, у процесі обміну інформацією, зміст даного регістра не змінюється;

– Регістр зберігання базового числа циклів прямого доступу до пам’яті;

– Регістр режима MR(Mode Register) визначає роботу канала.

Розглянемо адреси вказаних регістрів, які використовуються при програмування DMA.

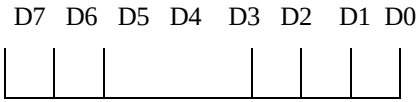
Порти 00h-07h: Ці регістри(порти) містять базові адреси і зміст лічильників передаваємих слів для каналів 0-3 . Їх призначення:

№ Порта	*Запис*	*Читання*	*№ Порта*	*Запис*	*Читання*
00h	Базова адреса канала 0	Поточна адреса відповідних каналів	01h	Лічильник канала 0	Поточне значення К-0
02h	Базова адреса канала 1		03h	Лічильник канала 1	Поточне значення К-1
04h	Базова адреса канала 2		05h	Лічильник канала 2	Поточне значення К-2
06h	Базова адреса канала 3		07h	Лічильник канала 3	Поточне значення К-3

Порт 08h

Цей порт використовується в якості керуючого регістру і при читання як регістр стану. Формат керуючого регістра розглянуто в попередній лекції.

Формат регістра стану



D0- D3 – встановлюються в 1 при досягненні лічильників відповідних каналів кінцевих значень.

D4- D7 – Встановлюються в 1, якщо маємо дозвіл для роботи відповідного канала.

Порт 09h

Регістр запиту. Призначений для організації програмного запиту на DMA . Для використання програмного запиту канал повинен бути запрограмований у режимі бітної передачі. Формат:

D7 D6 D5 D4 D3 D2 D1 D0

D0- D1 – номер канала

00 – канал 0

01 – канал 1

10 – канал 2

11 – канал 3

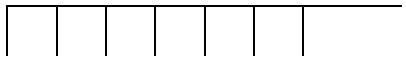
D2 – 0 – встановити програмний запит

1 – скинути програмний запит

D3- D7 – не використовуються

Порт 0Ah. Регістр маски.

D7 D6 D5 D4 D3 D2 D1 D0



D0- D1 – номер канала

00 – канал 0

01 – канал 1

10 – канал 2

11 – канал 3

D2 – 0 – встановити маску

1 – скинути маску

D3- D7 – не використовуються

Порт 0Bh. Регістр режиму. Формат в попередній лекції.

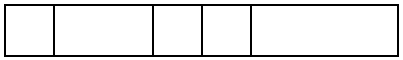
Порт 0Ch. Регістр скиду тригера байтів. Для завантаження внутрішніх 16-розрядних регістрів контролера використовується послідовний вивід молодшого, потім старшого байтів слова. Після скиду байтів, можна починати загрузку 16-розрядних регістрів.

Порт 0Dh. Регістр складу контролера. Після запису в цей регістр виникає скид контролера. Для подальшого використання контролер повинен бути заново проініціалізований.

Порт 0Eh. Регістр скиду маски. Після запису в цей регістр любого значення, дозволяється робота усіх 4-х каналів без масок.

Порт 0Fh. Регістр маскування/розмаскування потрібних каналів. Формат:

D7 D6 D5 D4 D3 D2 D1 D0



D0 – 0 –розмаскування каналу 0

1 – маскування каналу 0

D1 – 0 –розмаскування каналу 1

1 – маскування каналу 1

D2 – 0 –розмаскування каналу 2

1 – маскування каналу 2

D3 – 0 –розмаскування каналу 3

1 – маскування каналу 3

D4- D7 – не використовуються

Регістри сторінок (порти 81h-8fh)

Для роботи з пам’ятю контролер DMA використовує 20-розрядні фізичні адреси . 16 молодших бітів адрес необхідно записати в регістр базової адреси, а інші використати для адресації 16 сторінок. Таким чином реалізується 20-бітна базова адреса.

Для адресації регістрів сторінок для машин IBM PC XT використовуються порти:

81h –регістр сторінок канала 2;

82h – регістр сторінок канала 3;

83h –регістр сторінок канала 1;

87h – регістр сторінок канала 0;

Ініціалізація канала DMA

Для ініціалізації канала програма повинна виконати слідуючі дії:

1) Скинути тригер байтів командою запису в регістр 0Ch будь-якого значення;

2) Задати режим роботи каналу, виконавши запис за адресою 0Bh(у регістр режиму MR);

3) Записати молодші 16 бітів 20-бітової адреси області пам'яті, яка буде використана для передачі даних у регістр базової адреси;

4) Записати № сторінки (старші 4 біта 20-бітної адреси) у регістр сторінок;

5) Завантажити регістр циклів ПДП XWR значенням на 1 менше потрібної кількості передаваємих байтів;

6) Дозволити роботу канала, виконавши запис у регістр маски (0Ah);

Відразу після дозволу канал почне передачу даних.

Алгоритм програмування канала DMA

1. В регістр маски 0Ah внести 1 і замаскувати вибраний канал;
2. Скид тригера байтів – в порт 0Ch послати нулі;

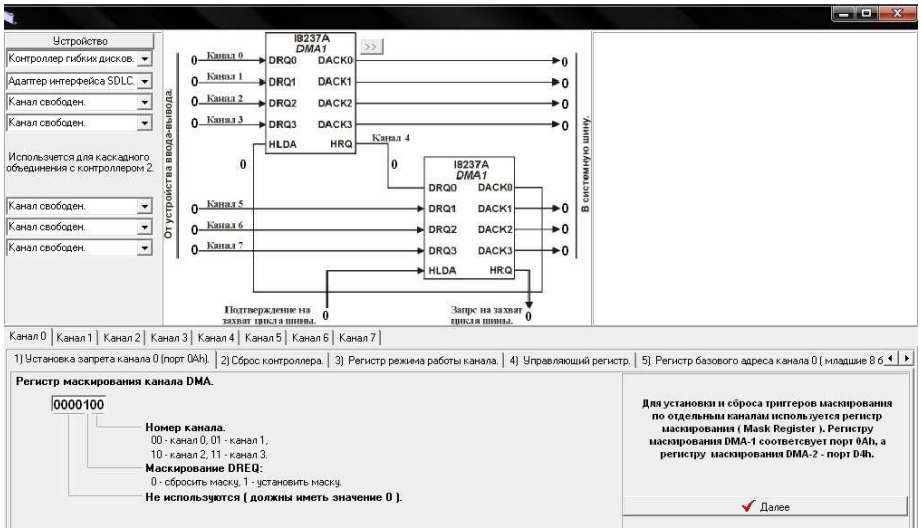
3. Вибрати необхідний режим каналу – регістр 0Bh;
4. За допомогою керуючого регістра 08h здійснити вибраний режим роботи;
5. Вводимо базову адресу відповідного каналу (відповідний байт);
6. Вводимо в регістр базової адреси старший байт;
7. Задати регістр сторінок (у D0 заносимо 1, всі останні – нулі);
8. Задати кількість циклів обміну в регістр 01h молодший байт;
9. Задати в регістр кількості циклів старший байт;
10. Скид регістру маски 0Ah, після чого контролер DMA починає роботу.

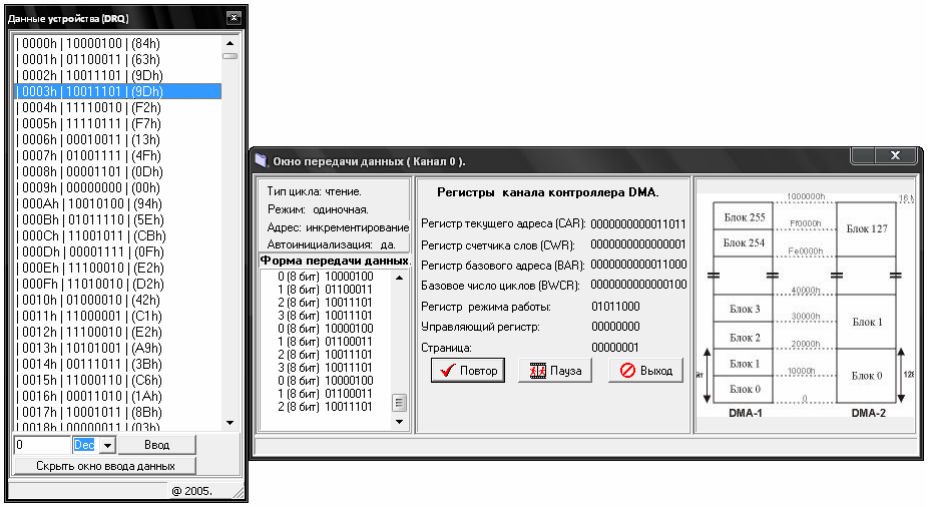
Програмування емулятора DMA

Для програмування DMA на емуляторі розглянемо 0-вий канал.

Алгоритм програмування емулятора:

1) Регістр маски 0Ah (00000000)
2) Скид тригера байтів 0Ch (00000000)
3) Регістр режиму канала 0Bh (01011000)
4) Керуючий регістр 08h
5) Регістр базової адреси канала 0h (00011000) (молодший байт)
6) Регістр базової адреси канала 0h (00000000) (старший байт)
7) В регістр сторінки посилаємо 1 (00000001)
8) Регістр базової кількості циклів 01h (00000100) (молодший байт)
9) Регістр базової кількості циклів 01h (00000000) (старший байт)
10) Скид маски 0Ah (00000000)





КПДП в IBM AT

DMA IBM AT складається із 2-х каскадно-включених мікросхем 8237A.

Другий контролер обслуговує канали DMA з № 4-7, які забезпечують 16-байтову передачу даних.

Використовується 8 бітів регістрів сторінок. Таким чином формується 24-бітна

адреса.

81h – регістр сторінок канала 2

82h – регістр сторінок канала 3

83h – регістр сторінок канала 1

87h – регістр сторінок канала 0

89h – регістр сторінок канала 6

8Bh – регістр сторінок канала 5

8Ah – регістр сторінок канала 7

Порти 0C/0h – порти 0c/0Eh. Ці регістри містять базові адреси і зміст лічильників передавємих даних каналів 4-7.

0C/0h – запис базової адреси канала 4/читиння базової адреси канала 4

0C/2h – запис лічильника канала 4/читиння лічильника канала 4

0C/4h – запис базової адреси канала 5/читиння базової адреси канала 5

0C/6h – запис лічильника канала 5/читиння лічильника канала 5

0C/8h – запис базової адреси канала 6/читиння базової адреси канала 6

0C/Ah – запис лічильника канала 6/читиння лічильника канала 6

0C/Ch – запис базової адреси канала 7/читиння базової адреси канала 7

0C/Eh – запис лічильника канала 7/читиння лічильника канала 7

Порти 0D0-0DEh

0D0h – керуючий регістр запису/керуючий регістр зчитування

0D2h – регістр запиту

0D4h - регістр маски

0D6h – регістр режима

0D8h – регістр скида тригера байтів

0DAh – скид контролера

0DCh – скид регістра маски

0DEh – маскування/розмаскування каналів

Регістр стану 008-0D0h:

Тимчасовий регістр 008-0D0h для зберігання даних під час передачі пам'ять-пам'ять завжди містить останній байт.

Приклад: Програмування канала 0 DMA на режим однократної передачі 100 байт за адресою 37856h

```
#include <dos.h>
void main ()
{
  long adr = 0x37856;
  outportb(0x0a,4);          // встановлення маски
  outportb(0x8,0);           // керуючий регістр
  outportb(0x0c,0);          // скид тригера
  ouportb(0x0b,0x54);        // 01010100bh - 54h
  outportb(0x00,adr&0xffff);
  outportb(0x83,adr>>16);
  outportb(0x01,99);         // 100 байт
  outportb(0x0a,0x0);        // скид маски
}
```


Література

1. Таненбаум Э.С. Архитектура компьютера. 5-е изд. – Спб.: Изд-во “Питер”, 2006. – 848 с.
2. Фролов А., Фролов Г. Аппаратное обеспечение IBM PC. / т. 2, книга 1. – М.: Диалог-МИФИ, 1992. – 208 с.
3. Голенкова Ж.К., Заблоцкий А.В., Мархасин Н.Л, Порошин С.С., Цесин Б.В., Шугаев А.Н. Руководство по архитектуре IBM PC AT. – Минск: “Консул”, 1992. – 949 с.
4. Гольденберг Л.М., Малев В.А., Малько Г.Б. Цифровые устройства и микропроцессорные системы. Задачи и упражнения: Учебн. пособие для вузов. – М.: “Радио и связь”, 1992. – 256 с.
5. Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. – М.: “Машиностроение”, 1989. – 263 с.
6. А. Фролов, Г. Фролов. Аппаратное обеспечение IBM PC. / т. 2, книга 2. – М.: Диалог-МИФИ, 1992. – 200 с.
7. И.В. Хмелевский, В.П. Битюцкий. Организация ЭВМ и систем. Однопроцессорные ЭВМ. Часть 3.: Конспект лекций / 2-е изд., испр. и допол. Екатеринбург: ГОУ ВПО УГТУ-УПИ, 2005. 100 с.
8. Журден Р. Справочник программиста на персональном компьютере фирмы IBM. – М.: “Радио и связь”, 1992. – 356 с.
9. Нортон П. Програмно-аппаратная организация компьютера IBM PC. Пер. с англ. Писарев С.Е. – К.: 1987. – 200 с.
10. Пей А. Сопряжение ПК с внешними устройствами. Пер. с англ. – М.: Изд-во “ДМК”, 2003. – 316 с.

Internet-підтрика курсу АЕОМ – **www.aeom.ho.ua**