**Лабораторная работа 1**

**Тема**: Работа с программируемым таймером i8253 / 54

**Цель**: Научиться программировать программируемый таймер ДЛЯ РАЗЛИЧНЫХ ИСПОЛЬЗОВАНИЯ В МП-системах

**Теоретические сведения**

Временные функции в микропроцессорных системах выполняют интегральные таймеры, которые представляют собой программируемые счетчики с предустановками и возможностью их программного чтения. Взаимодействуют таймеры с программой через систему прерываний и путем программного опроса. Типичным представителем таймера является микросхема i8253 и i8254.

Таймер выполнен nМоп-технологии, питается от источника напряжения +5V и потребляет ток 140 мА. Микросхема представляет собой многофункциональный трехканальный программируемый таймер. Каждый канал построен на основе 16-разрядного декрементирующего счетчика, работающего в двоичной или двоично-десятичной системе счисления с частотой счета до 2 МГц. Каждый счетчик программно можно настроить на один из шести режимов:

***- Режим 0 - прерывание по окончании счета;***

***- Режим 1 - формирование импульса программируемой длины;***

***- Режим 2 - генератор с программно-управляемой частотой;***

***- Режим 3 - делитель частоты;***

***- Режим 4 - формирователь строба по окончании счета;***

***- Режим 5 - счетчики событий.***

По правилам загрузки счетчика / таймера CE содержимым регистра CR все шесть режимов работы ПО можно разделить на три группы:

***1. Режимы 0, 4 - режимы однократного выполнения функций.***

D1

D2

D3

D4

D5

D6

D7

D8

CS

WR

RD

GATE0

GATE1

GATE2

PCI

A0

A1

CLK0

CLK1

CLK2

OUT0

OUT1

OUT2

+U

GND

***2. Режимы 1, 5 - режимы с перезапуском***

***3. Режимы 2, 3 - режимы автозагрузки.***

Содержание любого счетчика можно прочитать обычной операцией чтения или чтением "на лету" - без остановки счета.

**Назначение отдельных выводов микросхемы приведено ниже.**

CLK0, CLK1, CLK2 - счетные входы

OUT0, OUT1, OUT2 - выходы счетчиков

GATE0, GATE1, GATE2 - входы разрешения / запрета счета и аппаратного запуска счета

D0-D7 - линии шины данных

А0, А1 - линии адреса

RD, WR - сигналы чтения / записи

CS- сигнал выборки микросхемы.

Таймер программируется управляющим словом и установлением константы, определяющие параметры выполняемых функций. Регистр управляющего слова адресуется при А1, А0 = 11.

Формат управляющего слова:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SC1**  00-канал 0  01-канал 1  10-канал 2  11-код команды RBC  00-код команды CLS  01-чтения \ зап мл.байта к.  10-чтения \ зап ст.байту к.  11-чтения \ зап мл. а затем ст. байта к.  000-режим 0  001-режим 1  010-режим 2  011-режим 3  100-режим 4  101-режим 5  0-двоичный  1 Дв.\Дес счет | **SC0** | **RW1** | **RW0** | **M2** | **M1** | **M0** | **BCD** |

D1

D2

D3

D4

D5

D6

D7

D8

CS

WR

RD

GATE0

GATE1

GATE2

PCI

A0

A1

CLK0

CLK1

CLK2

OUT0

OUT1

OUT2

+U

GND

19

20

15

19

18

10

13

17

24

12

8

7

6

5

4

3

2

1

21

23

22

11

14

16

Управляющие слова могут записываться в любом порядке очередности выбора счетчика.

Очередность начальной загрузки может быть свободной. Загрузка счетчика обязательно должно осуществляться сразу после управляющего слова. Счет разрешается при высоком уровне на входе GATE. Результаты можно проконтролировать при чтении соответствующего счетчика.

При обычном чтении счетчика подсчет надо остановить (путем подачи на вход GATE = 0) или с помощью внешней логической схемы, блокирует прохождение импульсов на вход. Операцию чтения обязательно надо выполнять до конца, то есть если записано два байта константы, то читать обязательно надо именно два байта.

Режим чтения "на лету" позволяет прочитать содержимое счетчика без остановки счета. Для этого таймер необходимо запрограммировать на чтение.

Адреса портов:

PORT\_RUS 43h

PORT\_0 40h

PORT\_1 41h

PORT\_2 42h

ЗАДАЧИ (по сложности):

1. Используя программу «Proteus 7.10» создать систему: Процессор 8086;

шину адрес, шину данных и шину управления;

- Микросхема соответственно теме лабораторной (8253).

К выходам микросхемы можно подключить устройства управления (например светодиоды).

1. Написать программы программирования счетчика 0 таймера в режимах (на языке ASM или Си):

а) 0 и 1;

2. Определить слово состояния и содержание счетчика для всех трех каналов таймера.

3. Откомпилировать программы и запустит работу системы под управлением программы в «Proteus 7.10», или на программируемого комплексе из под компилятора «ASM 80» где

3.1. Отладить программы на

а) эмулятор-timer.exe

3.2. Снять осциллограммы сигналов с временными соотношениями на эмулятор.

4. Оформить отчет.

5. Запрограммировать один из каналов на перерыв через 10 мс где Тclk = 1 \ 1193180 c (сложность в, с).

6. Запрограммировать один из каналов на генерацию меандра с периодом 1мс ​​(сложность в, с).

7. сложить программу определения времени включения тумблеров (лаб.1).

**Контрольные вопросы.**

1. Назначение таймера.

2. Режимы работы.

3. Как перечислить необходимую частоту на выходе таймера в режиме генератора прямоугольных импульсов в байты для загрузки?

4. Можно запрограммировать различные счетчики на различные режимы?

5. Какой выход таймера надо использовать для организации прерывания через заданный промежуток времени и с чем его соединить?

Ответы на контрольные вопросы

1. Управление периферийными устройствами микропроцессорной системы, точного задания временных интервалов между управляющими сигналами.
2. Существует 6 режимов

Режим 0 - прерывание по окончании счета;

Режим 1 - формирование импульса программируемой длины;

Режим 2 - генератор с программно-управляемой частотой;

Режим 3 - делитель частоты;

Режим 4 - формирователь строба по окончании счета;

Режим 5 - счетчики событий.

1. Заданную частоту умножить на 2 и указать в константе в режиме №3
2. ДА. Можно каждый счетчик является независимым.
3. С выходом out

#include <stdio.h>

#include <dos.h>

void main () // главная подпрограмма

{

unsigned char const = 30; // константа

unsigned char register\_0= 0x30; // Слово управления

// Запись управляющего слова

outportb (0x43, register\_0)

// Запись константы

outportb (0x40, const & ff) // Запись младшего байта константы

outportb (0x40, const >> 8); // Запись старшего байта константы

exit (0);

}

#include <stdio.h>

#include <dos.h>

void main () // главная подпрограмма

{

unsigned char const = 8; // константа

unsigned char register\_0 = 0x32; // Слово управления

// Запись управляющего слова

outportb (0x43, register\_0)

// Запись константы

outportb (0x40, const & ff) // Запись младшего байта константы

outportb (0x40, const >> 8); // Запись старшего байта константы

exit (0);

}



