МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ПРИОРИТЕТНЫЕ НАЦИОНАЛЬНЫЕ ПРОЕКТЫ «Образование»

ПОБЕДИТЕЛЬ КОНКУРСА ИННОВАЦИОННЫХ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ ВУЗОВ

Ю.В. Китаев

ЛАБОРАТОРНЫЕ И ПРАКТИЧЕСКИЕ РАБОТЫ ”ЭЛЕКТРОНИКА И МП ТЕХНИКА” ЧАСТЬ 1

ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Санкт-Петербург
2008

УДК 681.32

Китаев Ю.В. Лабораторные и практические работы “Электроника и МП техника” / Ч.1. Учебное пособие. – СПб: СПбГУ ИТМО, 2008. – 90 с.

Приведены базовые лабораторные работы по типовым устройствам МП техники и схемотехнике цифровых устройств.

Для студентов, обучающихся по направлениям 200100 “Приборостроение” и 200200 “Оптотехника”

Рекомендовано к печати Советом ИФФ от 02 октября 2007г., протокол №2.

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

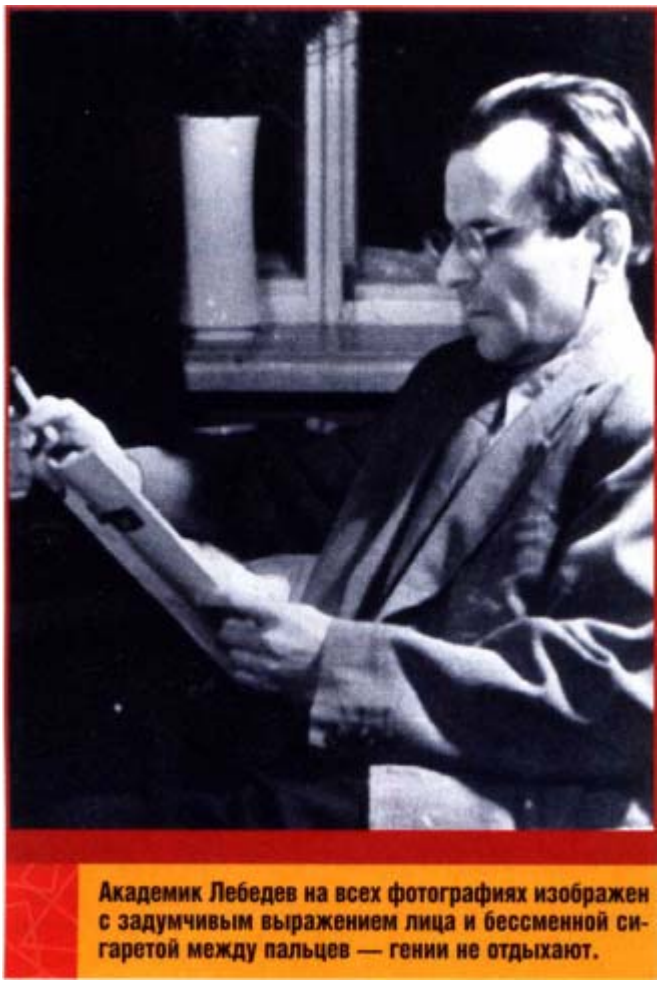
ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В первых двух разделах методического пособия приведены девять базовых лабораторных работ по двум разделам: микропроцессорная и цифровая техника. Приводятся основные методики расчета, проектирования и программирования простейших устройств. Основными инструментами в проведении лабораторных работ являются: моделирование устройств в среде САПР и отработка алгоритмов на рабочих стендах. Программирование ведется на ассемблере, как языке наиболее прозрачном для изучения взаимодействия программных и аппаратных средств.

Это дает возможность в лабораторных работах следующего цикла перейти от ассемблера к языкам высокого уровня, а от моделирования к непосредственному изготовлению прототипов разрабатываемых устройств.

В третьем разделе приведены примеры проектирования реальных устройств.



Академик Лебедев на всех фотографиях изображен с задумчивым выражением лица и бессменной сигаретой между пальцев — гении не отдыхают.

ЛАБОРАТОРНАЯ РАБОТА №1
РАЗРАБОТКА, ЗАПУСК И ОТЛАДКА ПРОГРАММ НА АССЕМБЛЕРЕ С ИСПОЛЬЗОВАНИЕМ УМК.

ЗАДАНИЕ: Написать две программы-утилиты:

1. Копирование содержимого одной области памяти, называемой - "ИСТОЧНИК" в другую область памяти, называемую - "ПРИЕМНИК",
2. Вычисление суммы содержимого заданной области памяти - "ИСТОЧНИК" (контрольная сумма).

Для выполнения заданий потребуются следующие 8-ми битовые программно-доступные регистры микропроцессора (МП) - A,B,..,L. Часть регистров может объединяться в двухбайтовые пары - BC, DE, HL. Регистр A называется также аккумулятором (рис. 1.1).

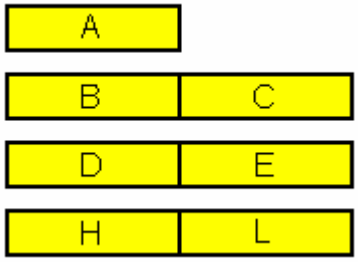


Рис. 1.1 Рабочие регистры МП

Схема алгоритма первой задачи.

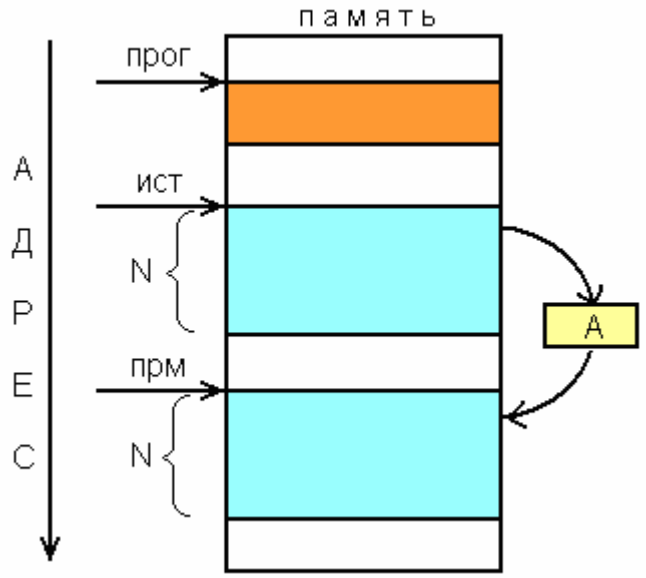


Рис. 1.2 Схема алгоритма первой задачи

На рисунке 1.2 приведена карта памяти, на которой "прог" - символический адрес начала программы копирования, "ист" и "прм" - символические адреса областей памяти "ИСТОЧНИК" и "ПРИЕМНИК".

N - число копируемых байтов (ячеек памяти). Копирование производится побайтно из текущей ячейки памяти "ИСТОЧНИКА" через регистр А микропроцессора в текущую ячейку памяти "ПРИЕМНИКА".

Блок - схема первой задачи (рис. 1.3).

Адреса текущих ячеек памяти "ИСТОЧНИКА" и "ПРИЕМНИКА" можно хранить в любых двухбайтовых регистрах, например BC и DE, а число копируемых байтов в одном из оставшихся, например в H.

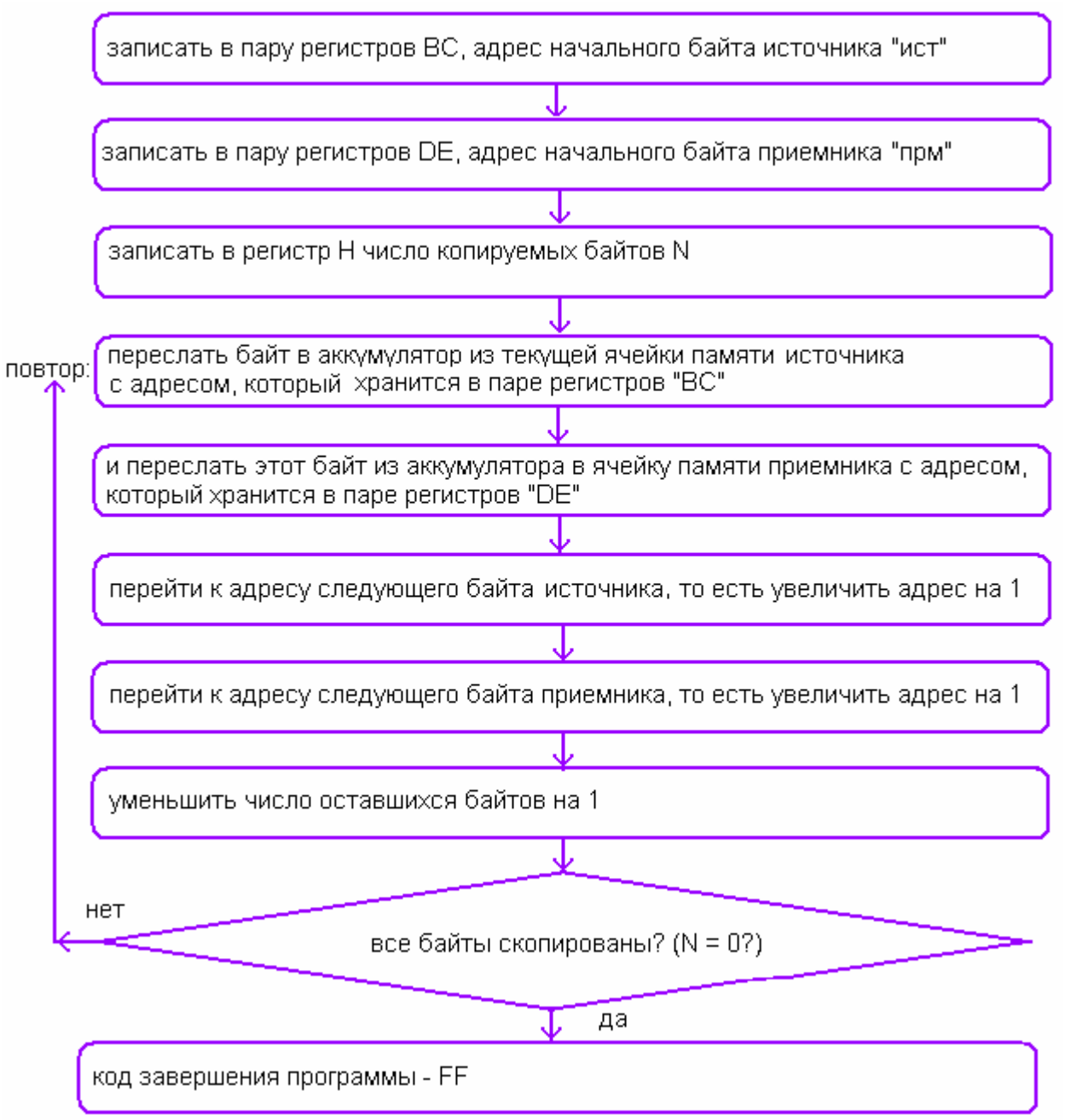


Рис. 1.3 Блок-схема первой задачи

Команды ассемблера, необходимые для решения первой задачи:

Каждому элементу блок - схемы соответствует одна команда. Команды отделены от описания (комментария) - ";". Сначала дается символическое описание команды. Порядок команд в списке не соответствует порядку операторов в блок-схеме.

LDAX B; (A) <-- ((BC)) - это означает, что байт из ячейки памяти с адресом, который хранится в паре регистров BC, пересылается (на самом деле копируется) в аккумулятор. Здесь и далее в символическом описании команды одинарные круглые скобки означают содержимое регистра. Двойные круглые скобки - содержимое ячейки памяти с адресом, который хранится в соответствующей паре регистров, в нашем случае в паре BC.

LXI D, прм; (DE) <-- прм . Записать в пару регистров DE двухбайтовое число с символическим именем "прм". При выполнении этой команды МП трактует "прм" просто, как двухбайтовое число, а не как адрес.

DCR H; (H) <-- (H) - 1. Уменьшить содержимое регистра H на 1. Такая команда называется - декремент.

LXI B, ист; (BC) <-- ист . Записать в пару регистров BC двухбайтовое число с символическим именем "ист".

STAX D; ((DE)) <-- (A). Переслать байт из аккумулятора в ячейку памяти с адресом, который хранится в паре регистров DE.

INX B; (BC) <--(BC) + 1. Увеличить содержимое пары BC на 1.

JNZ метка; Переход к физическому адресу с символическим именем "метка", если в результате, предшествующей команды, влияющей на "флаг нуля" получен не нулевой результат.

MVI H, N; (H) <-- N. Загрузить в регистр H однобайтовое число N.

INX D; (DE) <--(DE) + 1. Увеличить содержимое пары DE на 1.

Так может выглядеть решение 1-ой задачи, записанное на ассемблере MASM или TASM для семейства процессоров 80x86 (возможно это поможет вам при составлении своего варианта программы). Ваша запись, естественно, будет отличаться от этой.

```
        lea si, ист
        lea di, прм
        mov cl, 5
m1:     mov al, [si]
        mov [di], al
        inc si
        inc di
        dec cl
        jnz m1
```

Схема алгоритма второй задачи (рис. 1.4).

"ист" - символический адрес области памяти, содержимое которой (байты) суммируется. Окончательный результат (контрольная сумма) помещается в ячейку памяти, расположенную следом за последним слагаемым. Ячейка с суммой в общем случае может располагаться и по другому адресу. N - число байтов (слагаемых). Суммируется текущий байт из памяти с частичной суммой, хранящейся в аккумуляторе. Новая частичная сумма сохраняется в аккумуляторе (аккумулируется!). Конечный результат помещается в ячейку памяти СУММА.

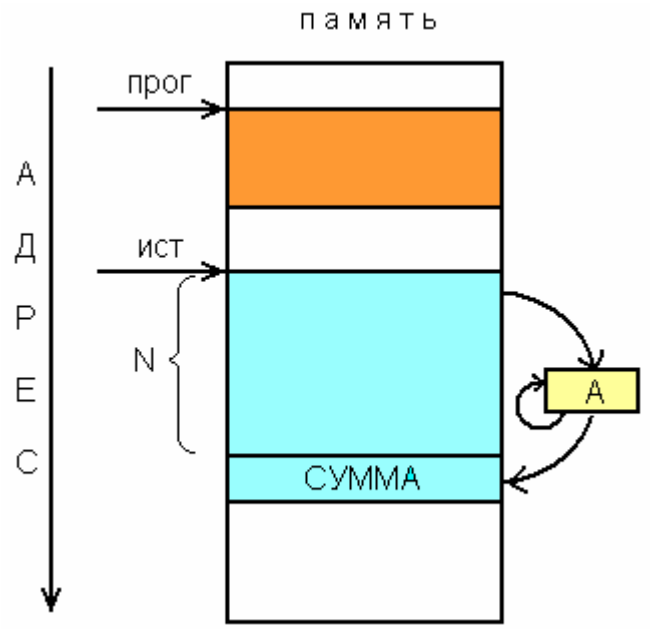


Рис. 1.4 Схема алгоритма второй задачи

Блок - схема второй задачи (рис. 1.5).

Адрес "ист", где хранятся слагаемые нужно записать ТОЛЬКО в пару регистров HL (потому что в командах сложения содержимое ячейки памяти - одно из слагаемых - адресуется только через пару HL). Число копируемых байтов поместим в одном из оставшихся регистров, например в C. В конце программы, как и в первой задаче, помещаем код FF (одна из команд прерывания). При выполнении этой команды микропроцессор прекращает выполнение пользовательской программы и возвращается к программе монитор.

ПРИМЕЧАНИЕ: Начальное слагаемое загружается в аккумулятор вне цикла "повтор", поэтому в регистр C помещаем число N -1

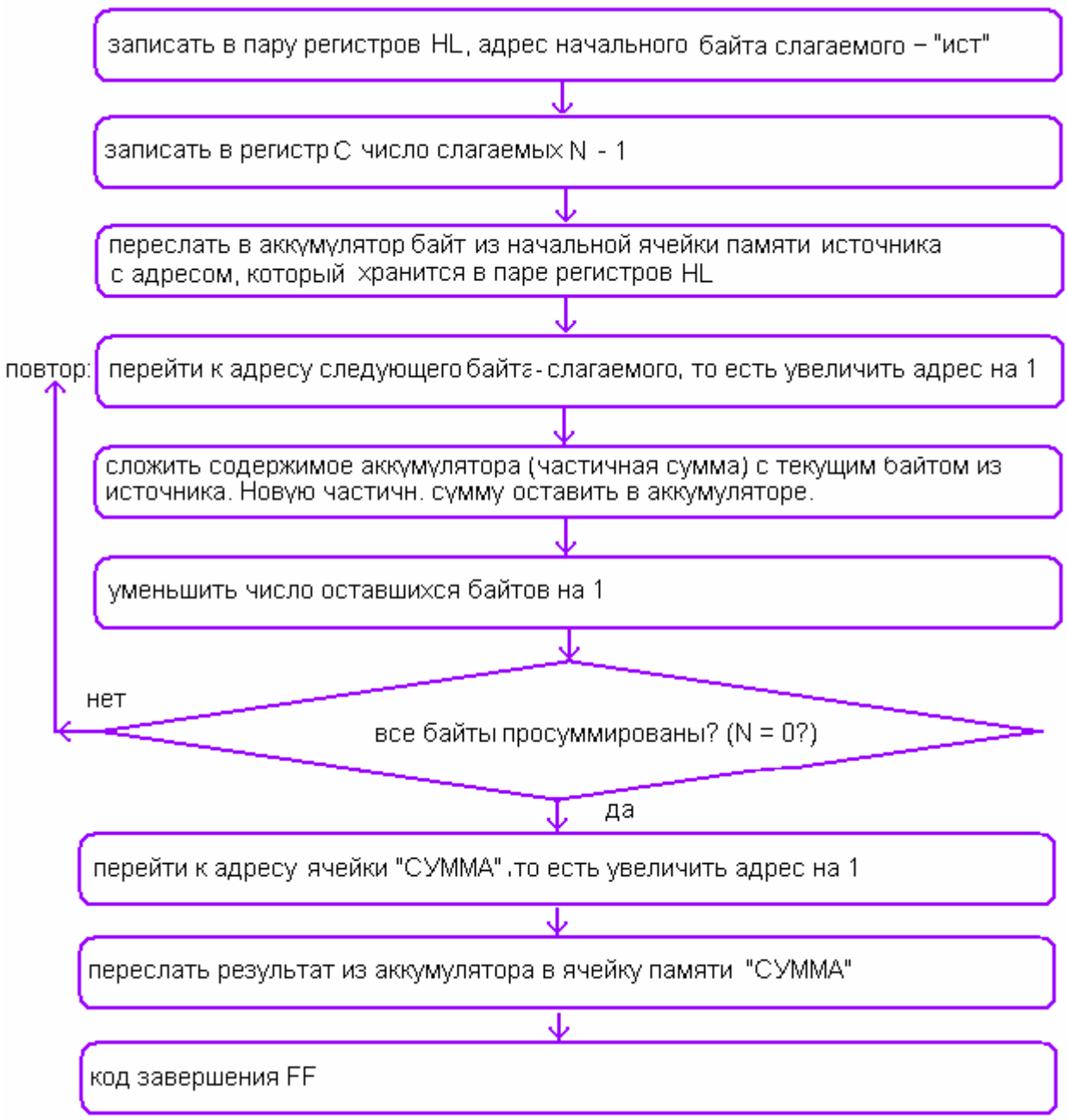


Рис. 1.5 Блок-схема второй задачи

Команды ассемблера, необходимые для решения второй задачи.

Каждому элементу блок - схемы соответствует одна команда. Команды отделены от описания (комментария) - ";". Сначала дается символическое описание команды. Порядок команд в списке не соответствует порядку операторов в блок-схеме.

MOV A, M; (A) <-- ((HL)) - байт из ячейки памяти с адресом, который хранится в паре регистров HL пересылается в аккумулятор. Здесь и далее операнд m обозначает содержимое ячейки памяти с адресом, который хранится в паре регистров HL. Почему эту команду Intel обозначила mov a,m а не ldax h по аналогии с ldax b и ldax d - остается загадкой.

ADD M; (A) <-- (A) + ((HL)). Сложить содержимое аккумулятора и ячейки памяти с адресом, который хранится в паре HL. Результат автоматически сохраняется в аккумуляторе.
MOV M, A; ((HL)) <-- (A). Переслать содержимое аккумулятора в ячейку памяти с адресом, который хранится в паре HL.
DCR C; (C) <-- (C) - 1. Уменьшить содержимое регистра C на 1.
LXI H, ист; (HL) <-- ист . Записать в пару регистров HL двухбайтовое число с символическим именем "ист". При выполнении этой команды МП трактует "ист" просто, как двухбайтовое число, а не как адрес.
JNZ метка; Переход к физическому адресу с символическим имененм "метка", если в результате, предшествующей команды, влияющей на "флаг нуля" получен не нулевой результат.
MVI C, N-1; (C) <-- N-1. Загрузить в регистр C однобайтовое число N-1.
INX H; (HL) <--(HL) + 1. Увеличить содержимое пары HL на 1.
Так может выглядеть решение 2-ой задачи, записанное на ассемблере MASM или TASM для семейства процессоров 80x86.

```
        mov bx,OFFSET ист
        mov cl,3 - 1
        mov al,[bx]
m1:     inc bx
        add al,[bx]
        dec cl
        jnz m1
        inc bx
        mov [bx],al
```

Пример записи программы (листинг программы):

ПРИМЕЧАНИЯ:
1. Команды могут быть одно-, двух- и трех-байтовыми,
2. Первым байтом любой команды является код операции или КОП, (в справочном приложении (таблица 1.2)) расположен слева от обозначения команды.
3. В таблице кодов (1.2) в двухбайтовых командах второй байт - операнд обозначен - d8 (восемь бит),
4. В таблице кодов (1.2) в трехбайтовых командах двухбайтовый операнд обозначен - d16 или Addr (16 бит).

Предположим в некой вымышленной программе первая команда - mvi c, 8 и начальный адрес программы прог = 0809(hex). Из справочника (таблица 1.2) находим код этой команды КОП = 0e(hex) и отмечаем, что команда двухбайтовая (d8). Поэтому в нашей программе эта команда занимает два байта по адресам 0809 и 080A. Здесь и далее КОП'ы выделены фиолетовым цветом.

Метка	Адрес	Код	Команда	Описание команды (комментарий)
прог	0809	0e	mvi c,08	команда пересылки (ваш текст здесь может быть более содержательным)
	080A	08		

Пусть вторая команда lxi b, 0924. Найдем ее КОП = 01. Команда трехбайтовая (d16), поэтому займет в памяти следующие три байта. Теперь листинг будет иметь следующий вид:

Метка	Адрес	Код	Команда	Описание команды (комментарий)
прог	0809	0e	mvi c,08	команда пересылки (ваш текст здесь может быть более содержательным)
	080A	08		
	080B	01	lxi b,**0924**	команда пересылки (ВНИМАНИЕ: байты операнда в трехбайтовой команде в колонке "код" **переставлены** - особенность Intel'овских процессоров)
	080C	**24**		
	080D	**09**		

Последними командами в нашем примере будет однобайтовая команда ldax d и трехбайтовая jnz m1, где m1 - метка. Предположим также, что в нашем примере переход должен производиться по адресу команды lxi b,0924. Окончательно листинг будет иметь вид (таблица 1.1):

Таблица 1.1

Метка	Адрес	Код	Команда	Описание команды (комментарий)
прог	0809	0e	mvi c,08	команда пересылки (ваш текст, здесь может быть более содержательным)
	080A	08		
m1	**080B**	01	lxi b,0924	команда пересылки (ВНИМАНИЕ: байты операнда в трехбайтовой команде в колонке "код" **переставлены** - особенность Intel'овских процессоров)
	080C	24		
	080D	09		
	080E	1a	ldax d	однобайтовая команда пересылки
	080F	c2	jnz m1	команда перехода (ВНИМАНИЕ: байты адреса в трехбайтовой команде в колонке "код" также **переставлены**). Символическому адресу "m1" соответствует физический адрес = **080B**.
	0810	**0B**		
	0811	**08**		
	...	...	...	...

Таблица 1.2 кодов операций некоторых команд (справочник).

Таблица 1.2

коды команд пересылок		коды арифм. команд и команд сравнения		коды логич. команд и команд сдвигов		коды команд переходов	
26	mvi h, d8	86	add m	07	rlc	C2	jnz Addr
0E	mvi c, d8	FE	cpi d8	0F	rrc	C3	jmp Addr
3E	mvi a, d8	25	dcr h				
16	mvi d,d8	0D	dcr c				
0A	ldax b	03	inx b				
1A	ldax d	13	inx d				
01	lxi b, d16	23	inx h				
11	lxi d, d16						
21	lxi h, d16						
12	stax d						
02	stax b						
7E	mov a, m						
77	mov m, a						
7A	mov a, d						
57	mov d, a						
DB	in d8						
D3	out d8						

Работа с лабораторным стендом - "Универсальный микропроцессорн. комплекс" в дальнейшем УМК.

1. Включение УМК.

а) нажать на клавишу  . На табло высветится "мусор" - неинициализированные данные.

б) перевести УМК в исходное состояние клавишей 

.

в) если УМК исправен, на левом индикаторе высветится "-" - символ готовности к вводу команд.

2. Запись байтов программы в память.

а) Нажать на клавишу  . Должны погаснуть все индикаторы.

б) Клавишами "01.. EF" набрать 4-х значный 16-ный адрес начала программы "прог" - в приведенном примере = 0809.

в) Нажать на клавишу "пробел"  . Высветится содержимое ячейки памяти по этому адресу.

г) Цифровыми клавишами набрать новое значение по этому адресу (например, по адресу 0809 нужно ввести 1e).

д) Повторять пп. (в,г) до последнего байта программы.

е) Завершить ввод кодов клавишей  .

3. Запись байтов данных в память.

Выполняется аналогично п.2, но в п.(б) вместо адреса "прог" нужно ввести адрес начала данных, например "ист" или "прм".

4. Просмотр содержиого памяти и/или исправление введенного кода.

В режиме исправления действия те же, что и п.2,3, но вместо п.(б) адреса "прог", "ист" и т.д. нужно набрать адрес неправильно введенного байта и затем исправить этот байт. В режиме просмотра (см. п.2,3) после ввода нужного адреса необходимо нажимать только на клавишу "пробел"  .

5. Запуск программы

а) Нажать на клавишу  . Должны погаснуть все индикаторы.

б) Клавишами "01.. EF" набрать 4-х значный 16-ный адрес начала программы "прог".

в) Завершить запуск клавишей  . Начинается выполнение программы.

ПРИМЕЧАНИЕ: Если на индикаторе высветится символ  - знак вопроса, это значит, что вы ошиблись и нажали не на ту директивную клавишу. Повторите ввод или нажмите на ту же директивную клавишу еще раз.

6. Оценка результата выполнения программы.

Если результат записывается в память, проверить его правильность можно, выполнив п.4. Например, в первой задаче содержимое области

памяти "прм" должно совпасть с содержимым "ист". Во второй задаче вычислите сумму и сравните ее с содержимым ячейки памяти "сумма".

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Получите варианты выполнения работы у преподавателя (физические адреса - "прог", "ист", "прм" и число байтов N). Допустимый диапазон адресов: 0800..0B80.
2. Для каждого элемента блок-схем рис.1.3 и 1.5 подобрать ЕДИНСТВЕННУЮ команду из приведенного перечня (см. разделы “команды ассемблера, необходимые для решения первой и второй задач”). Команды для каждой задачи записать в колонку и показать преподавателю.
3. Для каждой из 2-х программ заполните листинг (первые 4 колонки) по приведенному образцу (таблица 1.1). Коды команд берутся из таблицы 1.2. Каждый байт команды записывается на отдельной строке. Для определения числа байтов команды используйте 4-ре примечания к разделу “ Пример записи программы (листинг программы)”.
4. С разрешения преподавателя включите лабораторный стенд.
5. Введите в УМК из листинга коды программы и отдельно введите коды данных (массив байтов "ист").
6. Запустите программу (при необходимости исправьте ошибки) и предъявите результат преподавателю.

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ

1. Дать описание отдельных команд ассемблера, использующихся в программах.
2. Уметь комментировать каждую команду обеих программ.

ЛАБОРАТОРНАЯ РАБОТА №2
ВЫВОД ИНФОРМАЦИИ НА 8-МИ СЕГМЕНТНЫЙ ДИСПЛЕЙ

ЗАДАНИЕ: Рассчитать параметры схемы и написать программу вывода.

1. На основе приведенной схемы рассчитать ее основные параметры,
2. Написать программу.

Принципиальная схема установки приведена на рис. 2.1 (один из многих возможных вариантов).

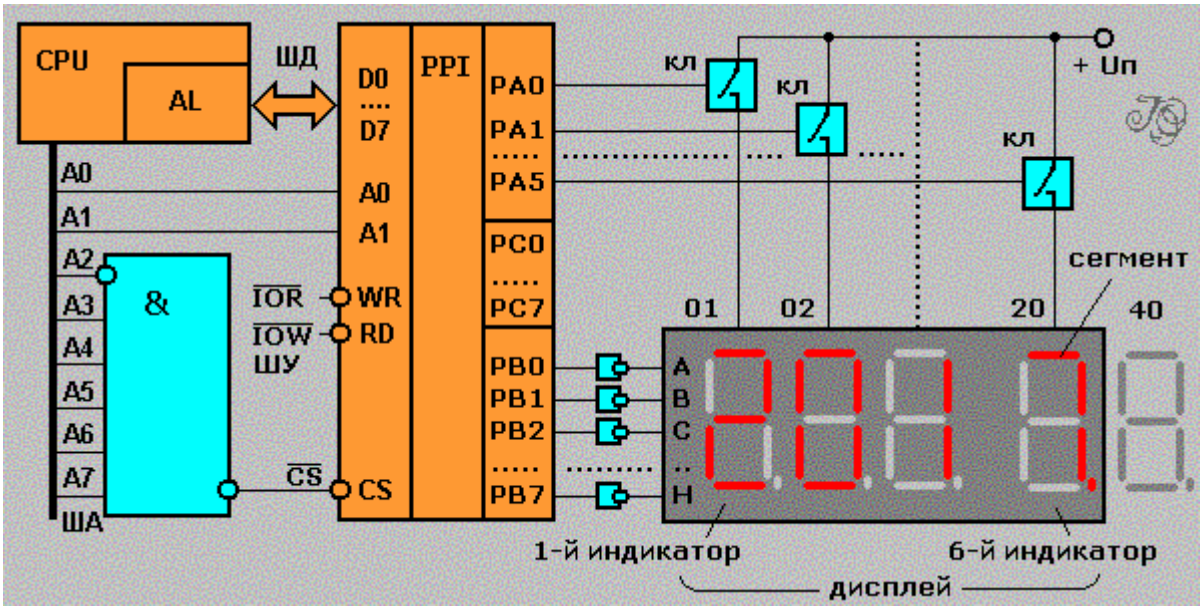


Рис.2.1 Схема вывода данных на 8-ми сегментный дисплей

Передача данных на 6-ти символьный 8-ми сегментный дисплей в приведенной схеме осуществляется с помощью микросхемы - программируемый периферийный интерфейс (ППИ, PPI). ППИ имеет три двунаправленных 8-ми битовых канала для обмена данными с микропроцессором, называемых также портами (PA, PB, PC). Сегменты дисплея могут быть, как жидкокристаллическими, так и светодиодными. В работе применяется светодиодный дисплей.

Слева от ППИ расположена стандартная интерфейсная схема подключения внешнего устройства (ВУ) - в нашем случае ППИ к микропроцессору (МП/CPU - Central Processing Unit). Справа от ППИ - функциональная часть схемы, обеспечивающая выполнение поставленной задачи.

Работа левой интерфейсной части схемы.

Любое программируемое внешнее устройство (в том числе ППИ) переводится в рабочий режим, т.е. подключается к шине данных (ШД), если на его вход "выбор микросхемы или выбор кристалла (Chip Select или Crystall Select)" подается активный сигнал ~CS=0(CS=1). В схеме инверсный вход ~CS соединен с выходом логического элемента "И-НЕ", выполняющего функцию неполного дешифратора адреса. По логике работы "И-НЕ", на его выходе будет "0", если на прямых входах единицы, а на инверсных нули. Отсюда нетрудно найти допустимые значения сигналов Ai на соответствующих линиях шины адреса (ША) для обращения микропроцессора к ППИ. Символами "Ф" обозначены пока не известные значения.

A7	A6	A5	A4	A3	A2	A1	A0
1	1	1	1	1	0	Ф	Ф

В схеме, линии ША A1,A0 подключены к соответствующим входам ППИ. Сигналы на этих входах (по справочнику) определяют выбор порта, через который будет производиться обмен данными между ППИ и МП. Значения на входах A1,A0 задаются в справочниках (табл. 2.1):

Таблица 2.1

A1	A0	порт / регистр управления
0	0	порт PA ППИ
0	1	порт PB ППИ
1	0	порт PC ППИ
1	1	порт регистра управления ППИ

Таким образом, сигналы на линиях шины адреса A7..A2 определяются разработчиком исходя из конкретной реализации дешифратора адреса, а значения A1,A0 выбираются из справочника. В таблице 2.2 даны рассчитанные значения адресов портов ППИ для приведенной КОНКРЕТНОЙ схемы подключения к МП. Содержимое регистра управления ППИ будет рассмотрено далее.

Таблица 2.2

A7	A6	A5	A4	A3	A2	A1	A0	16-ый адрес
1	1	1	1	1	0	0	0	= F8 Адрес порта PA
1	1	1	1	1	0	0	1	= F9 Адрес порта PB
1	1	1	1	1	0	1	0	= FA Адрес порта PC
1	1	1	1	1	0	1	1	= FB Адрес порта регистра управления

Обмен данными между ВУ(ППИ) и МП чаще всего осуществляется с помощью команд ассемблера: **in XX** и **out XX**, где XX - 16-ный адрес порта. Для нашей конкретной схемы: XX = (F8, F9, FA или FB). При выполнении команд in XX и out XX можно выделить 2 этапа:

1) микропроцессор помещает адрес XX на шину адреса. Этот адрес декодируется дешифратором в активный сигнал на одном из выходов (в схеме у дешифратора показан только один выход), который поступает на инверсный вход ~CS ВУ (в нашем случае ППИ). При этом ВУ(ППИ), как уже говорилось, переводится в рабочее состояние, т.е. подключается к ШД.

2) Второй этап имеет отличия. Команда **in XX**: МП генерирует на шине управления (ШУ) строб чтения ~IOR, которым байт данных считывается по ШД из адресуемого порта ВУ (в нашем случае ППИ) в аккумулятор МП. Команда **out XX**: МП генерирует на шине управления строб записи ~IOW, которым байт данных из аккумулятора микропроцессора записывается по ШД в выбранный порт ВУ (ППИ). Обычно этот байт защелкивается в триггерах порта.

Из описания выполнения команд in XX и out XX наглядно видно, как МП взаимодействует с ВУ при помощи 3-х шин.

Работа правой функциональной части схемы.

В общем случае информацию на дисплей можно выводить 2-мя основными способами: параллельным и последовательным.

В первом способе коды символов одновременно (параллельно) выводятся на соответствующие индикаторы. Напряжение питания на индикаторы дисплея также подается одновременно без электронных ключей. Достоинства: высокая скорость вывода и высокая видимая яркость индикаторов. Недостатки: повышенные аппаратурные затраты - на каждый индикатор нужен свой порт вывода и повышенное энергопотребление.

Во втором способе код текущего выводимого символа подается одновременно на все индикаторы через единственный порт, а напряжение питания подводится через электронный ключ только на выбранный индикатор. Достоинства: число портов резко сокращается, энергопотребление снижено. Недостатки: сравнительно невысокая видимая яркость и скорость вывода.

В данной работе используется последовательный способ вывода, причем код высвечиваемого символа выводится через порт PB, а напряжение питания на отдельный индикатор с помощью порта PA и электронных ключей. Вывод данных на дисплей можно производить в любом порядке, например, слева направо. Выведем сначала условия свечения отдельного светодиода (сегмента). На рисунке 2.2 приведена схема подключения к портам отдельного светодиода и его вольтамперная характеристика (ВАХ).

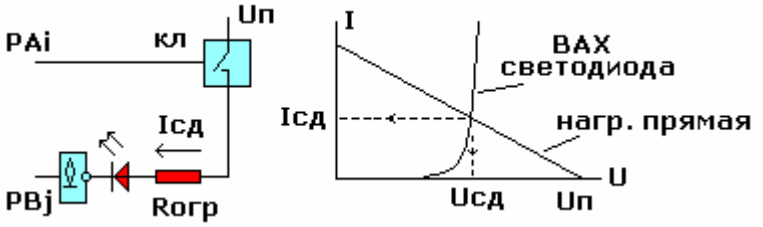


Рис.2.2 Подключение светодиода и его ВАХ

Для включения светодиода необходимо, чтобы его анод имел положительный потенциал относительно катода, т.е. необходимо замкнуть

контакты электронного ключа. Тогда напряжения питания +Uп через ограничивающий ток резистор поступит на анод светодиода, причем на катоде светодиода должен быть потенциал близкий к нулю. Электронные ключи могут быть двух типов: нормально замкнутые - когда для размыкания контактов необходимо подать управляющий сигнал и нормально разомкнутые, когда управляющий сигнал служит для замыкания контактов. В работе используются ключи с нормально разомкнутыми контактами. Т.е. для свечения необходимо на управляющий контакт ключа подать сигнал PAi = 1 (PAi - одна из линий порта PA). Катод светодиода через инвертор с открытым коллектором соединен с линией PBj порта PB. Поэтому, что бы на катоде был 0, на линии PBj должна быть 1.

Следовательно, условиями свечения будут равенства: PAi = PBj = 1, любая другая комбинация значений сигналов PAi и PBj будет "гасить" светодиод.

Например, чтобы выключить все светодиоды индикатора можно подать на их катоды высокий уровень сигнала, т.е. на все линии порта PB вывести нули. Этот прием далее будет использован в программе.

Попутно можно рассчитать значение ограничительного резистора - Rогр. Составим уравнение цепи: Uп = Uкл + Uгогр + Uсд + Uок = Uкл + Iсд*Rогр + Uсд + Uок. В этом уравнении величину Uп задает разработчик, например Uп = +5В. Напряжене Uок = Uлог.0 = 0.5В берется из справочника. Прямое падение напряжения на светодиоде, также из справочника находим равным Uсд =~1.8В. Величиной остаточного напряжения на контактах электронного ключа Uкл ~= 0.02В можно пренебречь. Прямой ток через светодиод Iсд, в справочнике называемый также номинальным Iном может быть различным для разных типов светодиодов. В примере Iсд = Iном примем равным 0.01А. Теперь можно найти Rогр = (5В - 1.8В - 0.5В)/0.01А = 370 Ом.

Теперь рассмотрим вывод кодов на дисплей подробнее (таблица 2.3). Начнем с крайнего левого индикатора. Для его активизации - крайний левый электронный ключ должен быть замкнут и напряжение питания Uп должно поступить ТОЛЬКО на этот индикатор, для других индикаторов ключи д.б. разомкнуты. Поэтому на линии PA0 должна быть "1", а на остальных выходах PA должны быть нули. Т.е. на выход порта PA программа должна поместить код, приведенный в первой строчке. Для вывода на следующий индикатор справа единица д.б. на линии PA1, а на остальных нули. И так далее, до последнего крайнего правого индикатора, у которого PA5 = 1. В результате, для последовательного перехода к следующему индикатору справа, необходимо сдвигать "1" на выходе порта PA влево.

Таблица 2.3

№	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	16-ный код
1	0	0	0	0	0	0	0	1	01
2	0	0	0	0	0	0	1	0	02
3	0	0	0	0	0	1	0	0	04
4	0	0	0	0	1	0	0	0	08
5	0	0	0	1	0	0	0	0	10
6	0	0	1	0	0	0	0	0	20
;-)	0	1	0	0	0	0	0	0	40

Такая смена кодов называется "бегущей единицей" (в других схемах последовательного вывода могут быть "бегущие нули"). Если сдвинуть единицу еще на один разряд влево - получим код 40(HEX) несуществующего (виртуального) индикатора справа от дисплея. Достижение этого кода будет признаком окончания вывода на дисплей. В правой колонке таблицы вычислены 16-ные коды на выходе порта PA, которые можно назвать также кодами индикаторов.

На этом описание и расчет правой функциональной части схемы закончены. Осталось написать программу. Однако, хотя в начале было сказано, что порты ППИ являются двунаправленными - это верно только отчасти. Например, в так называемом "нулевом режиме" порты являются квазидвунаправленными, и направление передачи данных через порт должен задавать программист.

Соглашение о направлении потоков данных. Во избежание путаницы все направления потоков информации в МП системе отсчитываются от микропроцессора, как показано на рисунке 2.3.

Рис. 2.3 Направление потоков данных

Обычно термины ввод-вывод относят к внешним устройствам, а слова запись-чтение к памяти, хотя это не принципиально. Термины передача-прием допустимы, но малоупотребительны. Направления потоков в ППИ задаются записью специального управляющего байта во внутренний регистр управления ППИ.

Формат управляющего байта ППИ в "режиме 0" (таблица 2.4). Программисту доступны только 4 его бита, в остальные биты записаны предопределенные значения. В справочнике определено правило - если

порт настраивается на ввод, то в соответствующий бит управляющего байта нужно заранее записать "1", ели на вывод - "0".

Таблица 2.4

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	PA	PCст	0	PB	PCмл

Особенностью порта PC является то, что его старшая и младшая половины настраиваются на ввод-вывод раздельно. Применительно к нашей схеме видно (и об этом уже говорилось), что порт PA настроен на ВЫВОД управляющих электронными ключами сигналов, поэтому бит PA = 0. Порт PB настроен на ВЫВОД кодов символов, поэтому бит PB также = 0. Порт PC в работе не используется, но в лабораторном стенде (УМК) к этому порту подключена клавиатура, следовательно биты PCмл и PCст должны быть установлны в "1". Таким образом, управляющий байт ППИ для нашей конкретной схемы равен 10001001(BIN) = 89(HEX).

Программа вывода информации на дисплей.

метка	адрес	код	команда	описание и/или комментарий
прог	0800	3E	mvi a, 89	записать управляющий байт ППИ в аккумулятор
	0801	89		
	0802	D3	out FB	и вывести его в регистр управления с адресом порта FB
	0803	FB		
снова	0804	01	lxi b, ист	поместить в пару регистров BC адрес начального кода выводимого символа ("ист" = 09A0)
	0805	A0		
	0806	09		
	0807	16	mvi d, 01	загрузить в регистр "d" код крайнего левого индикатора
	0808	01		
повтор	0809	7A	mov a, d	переслать из регистра в аккумулятор код текущего индикатора
	080A	D3	out F8	и вывести его в порт PA с адресом F8 (в этот момент напряжение Uп через электронный ключ поступит на текущий индикатор
	080B	F8		
	080C	0A	ldax b	переслать в аккум-р код символа из ячейки памяти с адресом в паре BC
	080D	D3	out F9	и вывести его в порт PB с адресом F9 (в этот момент, текущий символ высветится на соответствующем индикаторе)
	080E	F9		

	080F	3E	mvi a, 0	записать в аккумулятор нули
	0810	00		
	0811	D3	out F9	и вывести их в порт PB, т.е. "погасить" индикаторы. Реально "потухнет" только текущий. Для чего нужно гасить индикаторы смотри ниже в примечаниях.
	0812	F9		
	0813	03	inx b	перейти к адресу кода следующего выводимого символа
	0814	7A	mov a, d	сдвинуть содержимое регистра "d" на один бит влево, т.е. перейти к следующему индикатору справа (команда rlc производит циклический сдвиг аккумулятора на один бит влево, а rrc - вправо).
	0815	07	rlc	
	0816	57	mov d, a	
	0817	FE	cpi 40	сравнить текущий код индикатора в аккумуляторе с кодом 40(HEX) несуществующего индикатора (cpi N - сравнивает байт N с байтом в аккумуляторе)
	0818	40		
	0819	C2	jnz повтор	если коды не совпали, повторить вывод следующего символа в следующем индикаторе (команда jnz повтор - осуществляет условный переход к символическому адресу "повтор" = 0809
	081A	09		
	081B	08		
	081C	C3	jmp снова	если совпали, повторить вывод всех символов сначала (т.е. с крайнего левого индикатора). Команда jmp снова - безусловный переход к символическому адресу "снова" = 0804
	081D	04		
	081E	08		

Тот же код, записанный на ассемблере для МП семейства 80x86:

```
mov al,89h
out 0fbh,al
snova: mov bx, OFFSET istochnik
mov dl,1
povtor: mov al,dl
out 0f8h,al
mov al,[bx]
out 0f9h
mov al,0
out 0f9h
```

```
inc bx
rol dl,1
cmp dl, 40h
jnz povtor
jmp snova
```

ПРИМЕЧАНИЯ:

1. Гашение индикатора необходимо, иначе при подаче напряжения Uп на следующий индикатор справа (строка 080B) на нем на время выполнения команд (080C, 080D, 080E) будет высвечиваться символ из предыдущей позиции (т.к. код на выходе порта PB еще не сменился).
Новый символ на выходе порта PB появится на следующем индикаторе только после выполнения команды "out F9" (строка 080E). Зрительно это приводит к наложению двух символов (неправильному сочетанию светящихся сегментов) и невозможности прочитать выводимое сообщение.
2. Команда "jmp снова" создает "бесконечный" цикл, для того, чтобы выводимое сообщение можно было успеть прочитать. Выход из этого цикла, можно произвести, например вставкой дополнительного кода, отмечающего факт нажатия на некоторую клавишу. В лабораторном стенде из бесконечного цикла можно выйти, нажав на клавишу прерывания 

.
3. В существующих трансляторах ассемблера западных фирм символические имена должны содержать только символы ASCII (т.е. из нижней половины расширенной таблицы цифробуквенного кода). Поэтому метки "прог", "повтор", "снова", при использовании транслятора д.б. заменены, например на следующие "prog", "povtor", "snova".

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Получите вариант выполнения работы у преподавателя (физические адреса - "прог", "ист", число индикаторов и их начало, а также начертания выводимых символов). Допустимый диапазон адресов: 0800..0B80.
2. Введите изменения в листинг программы в соответствии со своим вариантом (начальный и конечный код индикаторов и т.д.).
3. С разрешения преподавателя включите лабораторный стенд.
4. Введите из листинга в УМК коды программы.
5. Рассчитайте и введите 8-ми сегментные коды выводимых символов. Каждому сегменту сопоставляется отдельный бит (рис.2.4). Причем обычно (а в нашем случае - так и есть) светящемуся сегменту соответствует "1".
Понятно, что с помощью такого кода записываются все цифры "012...9AbCdDEF", а также такие символы, как "П", "Р", "Г","L" и некоторые другие.

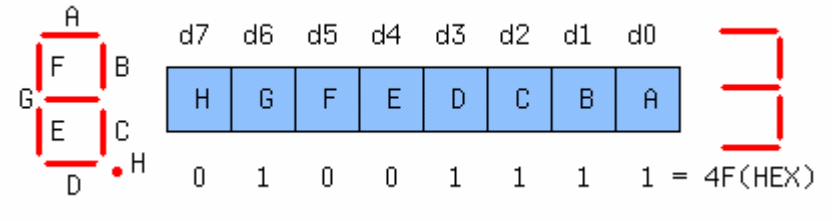


Рис. 2.4 Расчет 8-ми сегментного кода

5. Запустите программу (при необходимости исправьте ошибки) и предъявите результат преподавателю.

Вопросы для самопроверки:
1. Если переместить две команды гашения по адресам 080F..0812 (в таблице выделены цветом фона) между командами mvi d, 01 и mov a, d и привязать метку "повтор" к команде mvi a, 0, то видимая яркость дисплея возрастет. Почему?

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ И ЭКЗАМЕНА

0. Назначение, состав и принцип действия ППИ**.**
1. Объяснить работу схемы подключения ППИ к МП-системе (интерфейсная и функциональная части схемы).
2. Расчет адресов портов ввода/вывода.
3. Выполнение команд IN XX и OUT XX.
4. Способ вывода символов на дисплей. "Бегущий ноль". Для чего нужны электронные ключи и Rогр.?
5. Назначение отдельных битов управляющего байта ППИ.
6. Уметь комментировать каждую команду программы вывода на дисплей.

ЛАБОРАТОРНАЯ РАБОТА №3
РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ТАЙМЕРА С ПОДАЧЕЙ ЗВУКОВОГО СИГНАЛА И ФОРМИРОВАНИЕМ ЗВУКОВЫХ ОТМЕТОК

ЗАДАНИЕ: Рассчитать таймер и написать программу.

Теоретические и справочные сведения.

Для решения поставленной задачи воспользуемся микросхемой - Программируемый Интервальный Таймер (ПИТ). ПИТ содержит три 16-ти разрядных синхронных вычитающих счетчика с синхронной записью начального кода N (коэффициента деления или модуля счета). Здесь и далее могут использоваться термины "таймер" в смысле ПИТ и "таймер" в смысле соответствующего устройства для отсчета времени в названии практической работы. Различить эти термины нетрудно по их контексту.

Стандартная схема подключения ПИТ к МП системе приведена на рисунке 3.1.

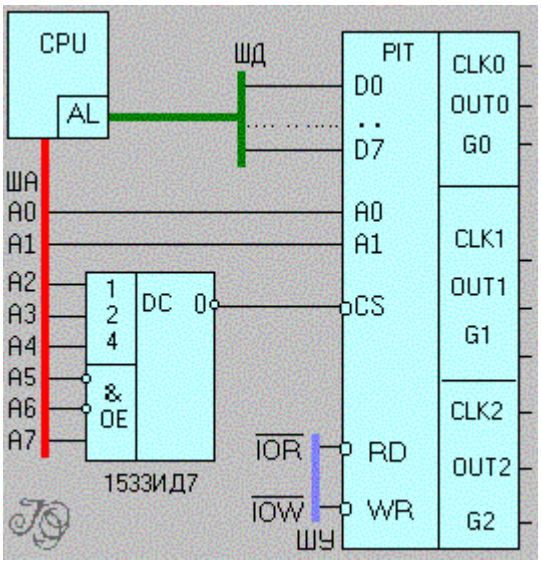


Рис. 3.1 Подключение ПИТ к МП системе

Каждый счетчик имеет три внешних вывода:
CLKi - вход сигнала (обычно, но не всегда - периодическая последовательность импульсов),
OUTi - выходной сигнал с заданными характеристиками (периодический или однократный),
Gi - вход сигнала управления запуском/остановом или разрешением/запретом счета.

Из условного обозначения ПИТ видно, что его левая часть имеет такие же обозначения выводов, как и у ППИ. Отличие только в назначении входов A1,A0. С помощью этих входов осуществляется выбор одного из 3-х счетчиков или внутреннего регистра управления ПИТ. Как и любая другая программируемая микросхема, имеющая вход ~CS(или CS), ПИТ подключается к ШД, если на этом входе активный уровень сигнала (в нашем случае ~CS = 0). На схеме этот вход соединен с нулевым выходом дешифратора, поэтому на его адресных входах должен быть двоичный код равный десятичному эквиваленту номера этого выхода, т.е. A4,A3,A2 = 000(BIN). На разрешающих входах OEi, которые объединены логической функцией "И" - "&", д.б. единицы на прямых входах и нули на инверсных. Т.е. A7,A6,A5 = 100(BIN). Две младшие линии ША подключены к одноименным входам A1,A0. В таблице 3.1 приведены справочные данные для этих входов.

Таблица 3.1

A1	**A0**	счетчик / регистр управления
0	0	счетчик СТ0 ПИТ

0	1	счетчик СТ1 ПИТ
1	0	счетчик СТ2 ПИТ
1	1	порт регистра управления ПИТ

В следующей таблице 3.2 даны рассчитанные значения адресов портов ПИТ для приведенной КОНКРЕТНОЙ схемы подключения ПИТ к МП.

Таблица 3.2

A7	A6	A5	A4	A3	A2	A1	A0	16-ый адрес
1	0	0	0	0	0	0	0	= 80 Адрес счечика СТ0
1	0	0	0	0	0	0	1	= 81 Адрес счечика СТ1
1	0	0	0	0	0	1	0	= 82 Адрес счечика СТ2
1	0	0	0	0	0	1	1	= 83 Адрес порта регистра управления

Обмен данными между ПИТ и МП также будет осуществляться с помощью команд ассемблера **in XX** и **out XX,** подробное описание которых приведено в работе №2.

Форма выходного сигнала, а также некоторые свойства счетчиков зависят от управляющих байтов, записываемых в регистры управления для каждого счетчика.

ФОРМАТ УПРАВЛЯЮЩЕГО БАЙТА ПИТ (таблица 3.3)

D7	D6	D5	D4	D3	D2	D1	D0

D7	D6	Номер счетчика		Способ загрузки Ni в счетчик	D5	D4
0	0	Счетчик 0 (СТ0)		Фиксация Ni	0	0
0	1	Счетчик 1 (СТ1)		Загрузка Ni одним младшим байтом	0	1
1	0	Счетчик 2 (СТ2)		Загрузка Ni одним старшим байтом	1	0
1	1	Запрет		Загрузка Ni двумя байтами	1	1

D3	D2	D1		Формат записи Ni в счетчик	D0
0	0	0	Режим 0	Запись Ni двоичным (16-ным) кодом	0
0	0	1	Режим 1	Запись Ni двоично-десятичным кодом	1

x	1	0	Режим 2
x	1	1	Режим 3
1	0	0	Режим 4
1	0	1	Режим 5

Два старших бита D7,D6 байта управления определяют номер регистра управления соответствующего счетчика. Как уже говорилось, адрес порта регистров управления счетчиками только один (A1,A0 = 11), а т.к. управляющих регистров три, по одному на каждый счетчик, то нужны дополнительные биты, идентифицирующие, какой управляющий байт к какому счетчику относится.

Биты D5,D4 задают способ записи начального значения Ni (коэффициента деления или модуля счета) i-го счетчика.
1) Если 3 <= Ni <=255, то начальное значение i-го счетчика МОЖНО программно загрузить одним младшим байтом, старший байт значение, которого равно 0 будет записан ПИТ'ом в счетчик автоматически (D5=0,D4=1).
2) Если 256 <= Ni $<2^{16}$ и кратно 256, то начальное значение i-го счетчика МОЖНО программно загрузить одним старшим байтом, младший байт значение, которого равно 0 будет записан ПИТ'ом в счетчик автоматически (D5=1,D4=0).
3) И, наконец, если 256 < Ni $<2^{16}$ и не кратно 256, то начальное значение i-го счетчика НУЖНО загружать двумя байтами (D5=1,D4=1).

Биты D3,D2,D1 определяют форму выходного сигнала. В режимах 0,2,3,4 счет разрешается при Gi=1 и запрещается при Gi=0. В режимах 1 и 5 для начала счета (запуск) на вход Gi необходимо подать положительный перепад.

Режим 0 (рис.3.2). Программируемая задержка времени. На выходе OUTi формируется положительный перепад через время Твых = Твх * (Ni + 1). Твх - период повторения импульсов на входе CLKi. Запуск счетчика CTi производится в момент записи в него Ni по команде out XX, при этом на входе Gi д.б. единица.

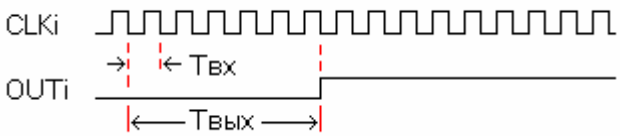


Рис.3.2 Режим 0

Режим 1 (рис.3.3). Генератор одиночного импульса заданной длительности. Длительность отрицательного импульса определяется следующим соотношением: Твых = Твх * Ni.Запуск счетчика CTi производится положительным перепадом на входе Gi.

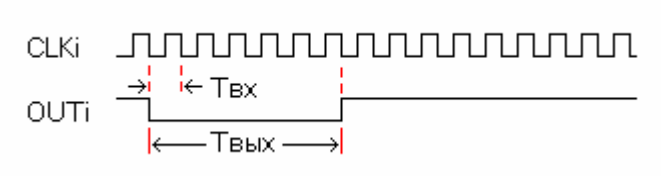


Рис.3.3 Режим 1

Режим 2 (рис.3.4). Генератор периодической последовательности импульсов со скважностью Q>2. Частота следования импульсов на выходе счетчика: Fвых = Fвх / Ni или Твых = Твх * Ni. Скважность - отношение периода повторения к длительности импульса (Твых/Ти). Запуск счетчика CTi производится в момент записи в него Ni по команде out XX, при этом на входе Gi д.б. единица.

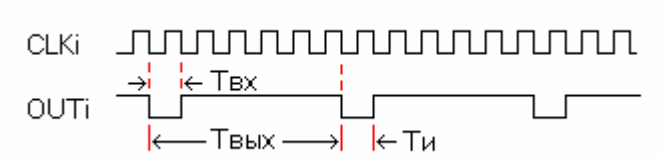


Рис.3.4 Режим 2

Режим 3 (рис.3.5). Генератор периодической последовательности импульсов со скважностью Q=2 (меандр). Частота следования импульсов на выходе счетчика такая же, как и во втором режиме Fвых = Fвх / Ni или Твых = Твх * Ni. Запуск счетчика CTi производится в момент записи в него Ni по команде out XX, при этом на входе Gi д.б. единица.

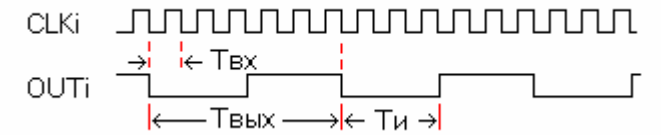


Рис.3.5 Режим 3

Режимы 4 и 5 (рис. 3.6). Программно и аппаратно управляемый строб-импульс. В режиме 4 формируется короткий импульс длительностью Твх с задержкой Твых = Твх*(N +1). А в режиме 5 формируется такой же импульс с задержкой Твых = Твх*N.

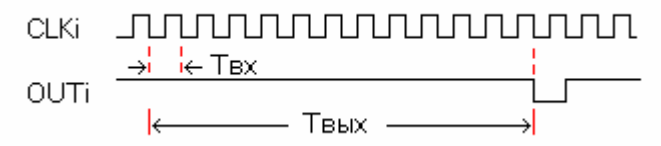


Рис.3.6 Режимы 4 и 5

Бит D0 регламентирует формат записи кода Ni в счетчик. Если D0=0, загружаемый код трактуется счетчиком, как двоичный (шестнадцатеричный) - в противном случае, как двоично-десятичный.

Принципиальная схема приведена на рис. 3.7 (один из многих возможных вариантов). Порядок подключения счетчиков в схеме 0->2->1 – произвольный.

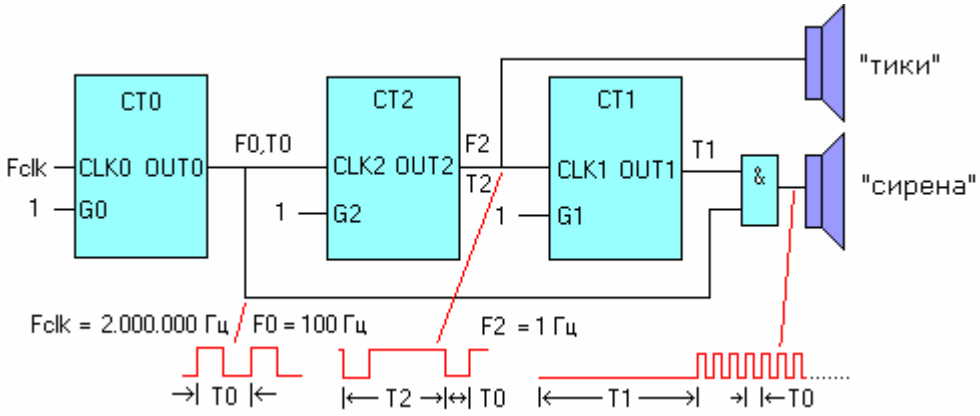


Рис. 3.7 Схема таймера с подачей звуковых сигналов

Исходные данные для проектирования (техническое задание):

1. Длительность интервала временной задержки T1 = 15сек.
2. Частота входного сигнала Fclk = 2МГц.
3. Частота звуковых отметок (тиков) F2 = 1Гц.
4. Частота сигнала (сирена) F0 = 100Гц.

Необходимо для каждого счетчика вычислить начальные значения Ni и управляющие байты.

Расчет счетчиков (рис. 3.7).

- Счетчик CT0. Биты управляющего байта D7,D6 = 00 для счетчика CT0. Для формирования на его выходе сигнала с частотой 100Гц подходят режимы 2 и 3. Однако в режиме 2 высокая скважность Q импульсов (длительность импульса мала по сравнению с периодом повторения) снижает мощность сигнала в Q/2 раз. Поэтому выбираем режим 3 и биты D3,D2,D1 = x11 = (011 или 111). Коэффициент деления CT0 в режиме 3 N0 = 2000000Гц / 100Гц = 20000(DEC) = 4E20(HEX). Так как 256 < N0(20000) $<2^{16}$ и не делится на 256, то загрузка кода N0 в счетчик должна производиться двумя байтами 4E и 20. Следовательно, биты D5,D4 = 11. Осталось найти значение бита D0. Двоично-десятичным кодом значение N0 = 20000 (пять десятичных цифр) невозможно загрузить в 16-ти битовый счетчик (четыре полубайта - четыре десятичных цифры), поэтому D0 = 0 (загрузка N0 в двоичном коде). Таким образом, управляющий байт для CT0 будет равен 36(3E):

D7	D6	D5	D4	D3	D2	D1	D0	HEX код
0	0	1	1	0	1	1	0	36
0	0	1	1	1	1	1	0	3E

- Счетчик CT2. Биты управляющего байта счетчика CT2 D7,D6 = 10. Для формирования на его выходе сигнала с частотой 1Гц также подходят режимы 2 и 3. Однако, в режиме 3 при скважности Q=2 пьезокерамические пластины верхнего динамика будут издавать звук при каждом перепаде на выходе CT2, т.е. отметки времени будут звучать с двойной частотой, что естественно не планировалось. Поэтому выбираем режим 2 и биты управляющего байта D3,D2,D1 = x10 = (010 или 110). Модуль счета N2 счетчика CT2 равен F0 / F2 = 100(DEC) = 64(HEX) и находится в пределах 3<=N2<=255. Из этого следует, что N2 можно записать в счетчик одним младшим байтом, т.е. биты D5,D4 = 01. Однобайтовое значение N2 содержит две тетрады, в которых невозможно разместить три десятичных цифры, поэтому запись N2 должна производиться не двоично-десятичным кодом, а двоичным и бит D0 = 0. Суммируя сказанное, получим два возможных значения управляющего байта CT2.

D7	D6	D5	D4	D3	D2	D1	D0	HEX код
1	0	0	1	0	1	0	0	94
1	0	0	1	1	1	0	0	9C

- Счетчик CT1. Биты управляющего байта счетчика CT2 D7,D6 = 01. Временная задержка на его выходе может быть сформирована режимами 0 или 1. В режиме 1 требуется дополнительный генератор импульса для запуска счетчика положительным перепадом на входе G1. В режиме 0 дополнительный генератор не нужен, поэтому остановимся на этом режиме. В режиме 0 для работы счетчика на входе G1 должен быть сигнал = 1 и следовательно биты D3,D2,D1 = 000. Коэффициент деления в режиме 0: N1 = (T1 / T2) - 1 = 15/1 -1 = 14(DEC) = E(HEX). Это число также может быть загружено в CT1 одним младшим байтом, поэтому биты D5,D4 = 01. Модуль счета N1 = 14 можно записать, либо шестнадцатиричным кодом 0E(00001110 BIN), либо двоично-десятичным кодом 14(00010100 BCD). Это число (временная задержка) является тем параметром, который потребитель самостоятельно вводит в таймер (с панели управления прибором) и знание шестнадцатиричной системы счисления не входит в круг его обязанностей. Поэтому разработчику таймера нужно предусмотреть установку пользователем числа N1 десятичными цифрами и

следовательно бит D0 управляющего байта равен 1. Окончательно, управляющий байт для CT1 будет равен:

D7	D6	D5	D4	D3	D2	D1	D0	HEX код
0	1	0	1	0	0	0	1	51

Программа запуска приведенной схемы таймера:

```
;#### подготовка счетчиков таймера к работе в выбранных режимах
mov al,3eh  ;  загрузка управляющего байта CT0
out 83h,al;  по адресу 83
mov al,94h ; загрузка управляющего байта CT2
out 83h,al; по адресу 83
mov al,51h; загрузка управляющего байта CT1
out 83h,al; по адресу 83
;#### запись коэффициентов деления  Ni в счетчики
;#### в режимах 0,2,3 счетчики при записи в них последнего
;#### (или единственного) байта Ni начинают счет импульсов CLKi
mov al,20h  ; запись младшего байта N0=20000(DEC)=4e20(HEX)
out 80h,al; в счетчик CT0 по адресу 80
mov al,4eh  ; а теперь - старшего байта N0
out 80h,al; по тому же адресу
mov al,64h  ; запись N2=100(DEC) в счетчик CT2
out 82h,al; по адресу 82
mov al,14h  ; запись N1=14(BCD !) в счетчик CT1
out 81h,al; по адресу 81
;#### таймер начал отсчет времени.
```

ПРИМЕЧАНИЕ: несмотря на то, что в команде "mov al,14h" операнд записан, как и все остальные операнды в программе - в 16-ном коде (об этом в частности свидетельствует окончание "h"), ПИТ'ом это число 14 будет истолковано, не как 16-ное (с десятичным эквивалентом 16+4=20, а как двоично-десятичное (10+4=14)!

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Получите вариант выполнения работы у преподавателя (физический адрес - "прог", значения частот: Fclk, F0, F2, значение задержки - T1 и номер выхода дешифратора).
2. Введите изменения в программу и заполните листинг в соответствии со своим вариантом.
3. Потребуйте у преподавателя подключить плату расширения с ПИТ и включить лабораторный стенд.
4. Введите из листинга в УМК коды программы.

5. Запустите программу (при необходимости исправьте ошибки). На слух или по секундомеру оцените работу программы и предъявите результат преподавателю.

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ И ЭКЗАМЕНА

0. Назначение, состав и принцип действия ПИТ**.**
1. Объяснить работу схемы подключения ПИТ к МП-системе (интерфейсная и функциональная части схемы).
2. Расчет адресов портов ввода/вывода.
3. Назначение отдельных битов управляющего байта и режимы работы ПИТ.
4. Схема таймера для обратного отсчета времени с подачей звуковых отметок.
5. Расчет управляющих байтов и коэффициентов деления Ni каждого счетчика.
6. Уметь комментировать каждую команду программы запуска таймера.

ЛАБОРАТОРНАЯ РАБОТА №4
ПРОГРАММА-МОНИТОР И ПРОСТАЯ КОНСОЛЬ

Цель работы - изучение аппаратных и программных средств организации диалогового режима работы оператора и микроЭВМ с использованием простой консоли.

Монитором называется программа, резидентно (постоянно) находящаяся в ПЗУ и позволяющая вводить в память коды пользовательских программ, а также редактировать, отлаживать и выполнять их, т.е. монитор является простейшей операционной системой. Консоль - историческое название клавиатуры и дисплея.

Приведенная далее программа монитор написана для консоли содержащей 1) три линейки из восьми светодиодов для отображении 16-ного адреса и содержимого ячейки памяти по этому адресу, 2) восемь клавиш: 4 - директивных (MSBA, LSBA, MOD, PROG) и 4 - цифровых (0, 1, 2, 3).

Директивные клавиши SW7..SW4 служат: SW4 - для ввода старшего байта адреса "MSBA - Most significant byte address", SW5 - для ввода младшего байта адреса "LSBA - least significant byte address", SW6 - для ввода байта данных по указанному адресу и перехода к следующей ячейке памяти "MOD", SW7 - для пуска пользовательской программы, начиная с заданного адреса "PROG". Оставшиеся 4 клавиши можно использовать либо для ввода двоичного кода (две клавиши), оставив две под резерв, либо для ввода кода по основанию 4, т.е. четверичного кода или тетракода. Тетракод получается из двоичного кода путем разбиения последнего на пары двоичных цифр и суммирования весовых коэффициентов. Например, 11100100(BIN) = 11 10 01 00 = 3210(тетракод). Выбираем тетракод,

позволяющий ускорить ввод в два раза. Указанных 8-ми клавиш достаточно для решения любых задач.

Один из возможных способов подключения консоли к МП с помощью ППИ показан на рисунке 4.1. Стандартная левая интерфейсная часть схемы подключения ППИ к МП на рисунке не показана (см. лабораторную работу №2).

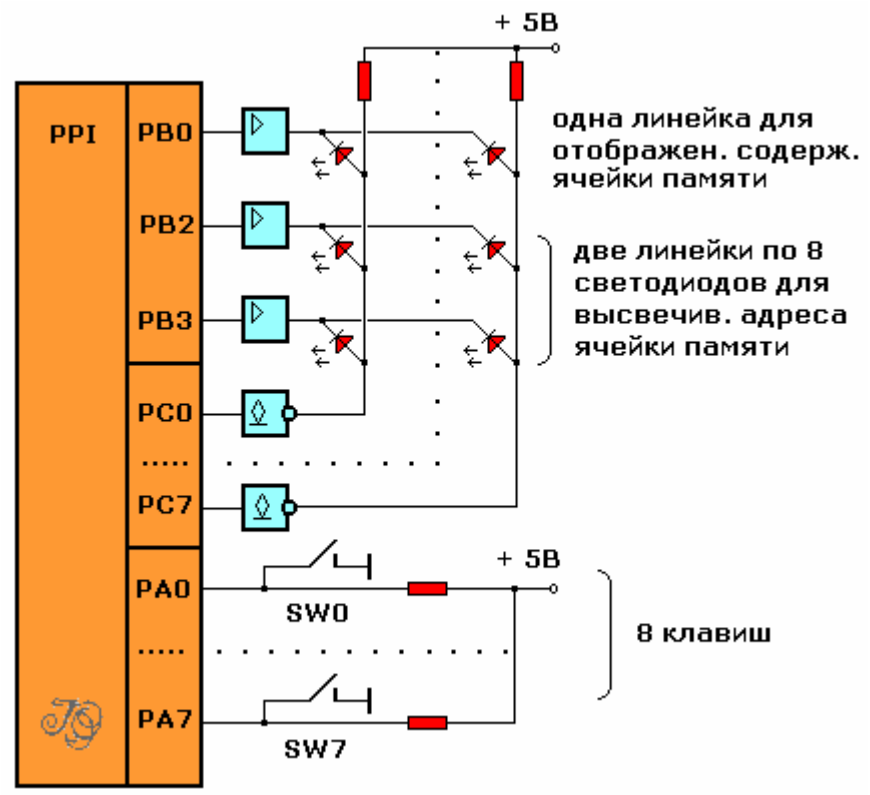


Рис. 4.1 Схема подключения консоли

Клавиатура подключена к порту PA. Если не нажата ни одна из клавиш, то на всех входах PA - высокий уровень сигнала +5В (все "1" - код FF). При нажатии, например на клавишу SW2 только на входе PA2 будет "0" (поэтому сканкод клавиши SW2 = 1111 1011 = FB).

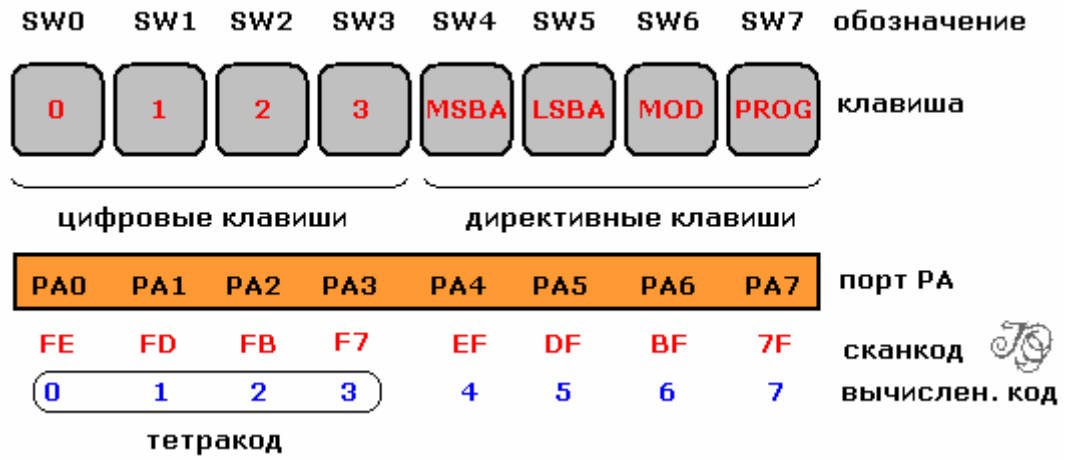


Рис. 4.2 Назначение клавиш и светодиодов консоли

Назначение клавиш: MSBA - ввод старшего байта адреса, LSBA - ввод младшего байта адреса, MOD - модификация содержимого ячейки памяти с текущим адресом и увеличение адреса на 1, PROG - запуск программы пользователя с указанного (высвеченного) адреса.

Выбор линейки осуществляется выводом логического нуля в соответствующий разряд порта PB при этом на катоды светодиодов выбранной линейки, с выхода логического элемента с повышенной нагрузочной способностью подается близкий к нулю потенциал.

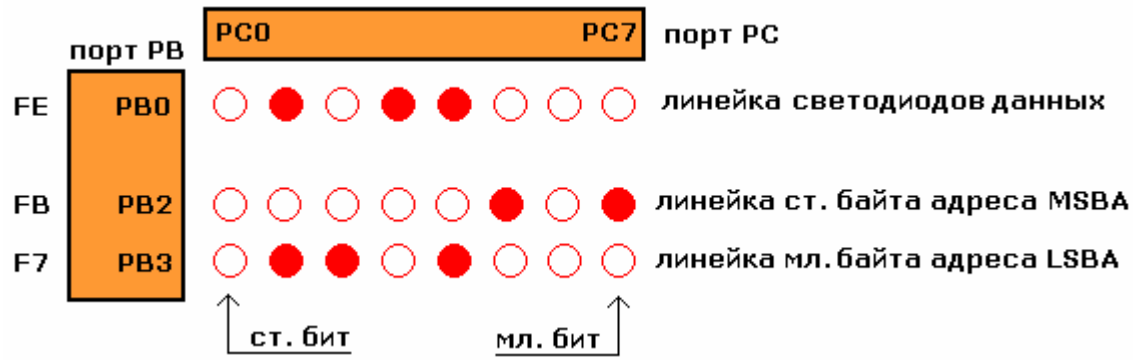


Рис.4.3 Пример вывода кода

Например, для отображения младшего байта адреса необходимо на катоды светодиодов подать нули, т.е. на линию PB3 вывести 0, а на остальные линии порта PB - единицы (1111 0111 = F7). Аналогично код линейки старшего байта адреса = 1111 1011 = FB, а код линейки данных = FE.

Двоичный код (байт адреса или байт данных) в инверсном виде выводится через порт PC и через инверторы с открытым коллектором поступает на аноды светодиодов. На рисунке высвечивается код 58(HEX) по адресу 0568(HEX). Следует также отметить, что младший бит порта PC подключен к левым, т.е. старшим битам линеек. Поэтому для правильного считывания высвеченного на линейках кода потребуется подпрограмма перестановки битов.

Характер информации, высвечиваемой на линейке данных, зависит от того, какая клавиша была нажата последней. Если нажата:

а) одна из клавиш адреса MSBA или LSBA, то высветится содержимое ячейки памяти с НОВЫМ адресом (HL);
б) клавиша модификации MOD содержимого текущей ячейки памяти - высветится содержимое ячейки памяти с адресом (HL)+1;
в) одна из цифровых клавиш - индицируется содержимое регистра C, т.е. набираемый код.

Пример отображения информации на светодиодных линейках при работе с программой монитор.

1. Начало работы монитора: ● - светодиод "горит" (бит = 1), ○ - светодиод погашен (бит = 0), ? - светодиод может быть в любом состоянии (бит = 1 или 0).

D7	D6	D5	D4	D3	D2	D1	D0	код	начальное неинициализированное значение в ячейке памяти (мусор) с адресом 0A00, регистр C
?	?	?	?	?	?	?	?	(C)	
A15	A14	A13	A12	A11	A10	A9	A8	адрес	начальный адрес 0A00 = 0000 1010 0000 0000 в паре регистров HL
○	○	○	○	●	○	●	○	(H)	
○	○	○	○	○	○	○	○	(L)	
A7	A6	A5	A4	A3	A2	A1	A0	адрес	

Фрагмент карты памяти в этот момент имеет следующий вид:

адрес	код							
0A00	?	?	?	?	?	?	?	?
0A01	?	?	?	?	?	?	?	?

2. Ввод в регистр C числа 8D(HEX) = 10 00 11 01(BIN) = 2031(тетракод):

D7	D6	D5	D4	D3	D2	D1	D0	код	нажата цифровая клавиша "2"
?	?	?	?	?	?	●	○	(C)	
?	?	?	?	●	○	○	○	(C)	нажата цифровая клавиша "0"
?	?	●	○	○	○	●	●	(C)	нажата цифровая клавиша "3"
●	○	○	○	●	●	○	●	(C)	нажата цифровая клавиша "1"
A15	A14	A13	A12	A11	A10	A9	A8	адрес	начальный адрес 0A00 = 0000 1010 0000 0000 в паре регистров HL
○	○	○	○	●	○	●	○	(H)	
○	○	○	○	○	○	○	○	(L)	
A7	A6	A5	A4	A3	A2	A1	A0	адрес	

3. Ввести число 8D = 1000 1101 в ячейку памяти с адресом 0A00, нажав на клавишу MOD:

D7	D6	D5	D4	D3	D2	D1	D0	код	начальное
?	?	?	?	?	?	?	?	(C)	неинициализированное значение в ячейке памяти (мусор) с адресом 0A01, регистр C
A15	A14	A13	A12	A11	A10	A9	A8	адрес	адрес следующей ячейки 0A01 = 0000 1010 0000 0001 в паре регистров HL
○	○	○	○	●	○	●	○	(H)	
○	○	○	○	○	○	○	●	(L)	
A7	A6	A5	A4	A3	A2	A1	A0	адрес	

Фрагмент карты памяти в этот момент имеет следующий вид (код 8D помещен в ячейку 0A00):

адрес	код							
0A00	1	0	0	0	1	1	0	1
0A01	?	?	?	?	?	?	?	?

4. Ввод в регистр C кода 56(HEX) = 01 01 01 10(BIN) = 1112(тетракод): аналогично п.2.

5. Ввести код 56(HEX) в младший байт адреса, нажав на клавишу LSBA:

D7	D6	D5	D4	D3	D2	D1	D0	код	начальное
?	?	?	?	?	?	?	?	(C)	неинициализированное значение в ячейке памяти с адресом 0A56, регистр C
A15	A14	A13	A12	A11	A10	A9	A8	адрес	адрес ячейки 0A56 = 0000 1010 0101 0110 в паре регистров HL
○	○	○	○	●	○	●	○	(H)	
○	●	○	●	○	●	●	○	(L)	
A7	A6	A5	A4	A3	A2	A1	A0	адрес	

6. и т.д.

ПРОГРАММА МОНИТОР И ЕЕ БЛОК СХЕМА (рис. 4.4).

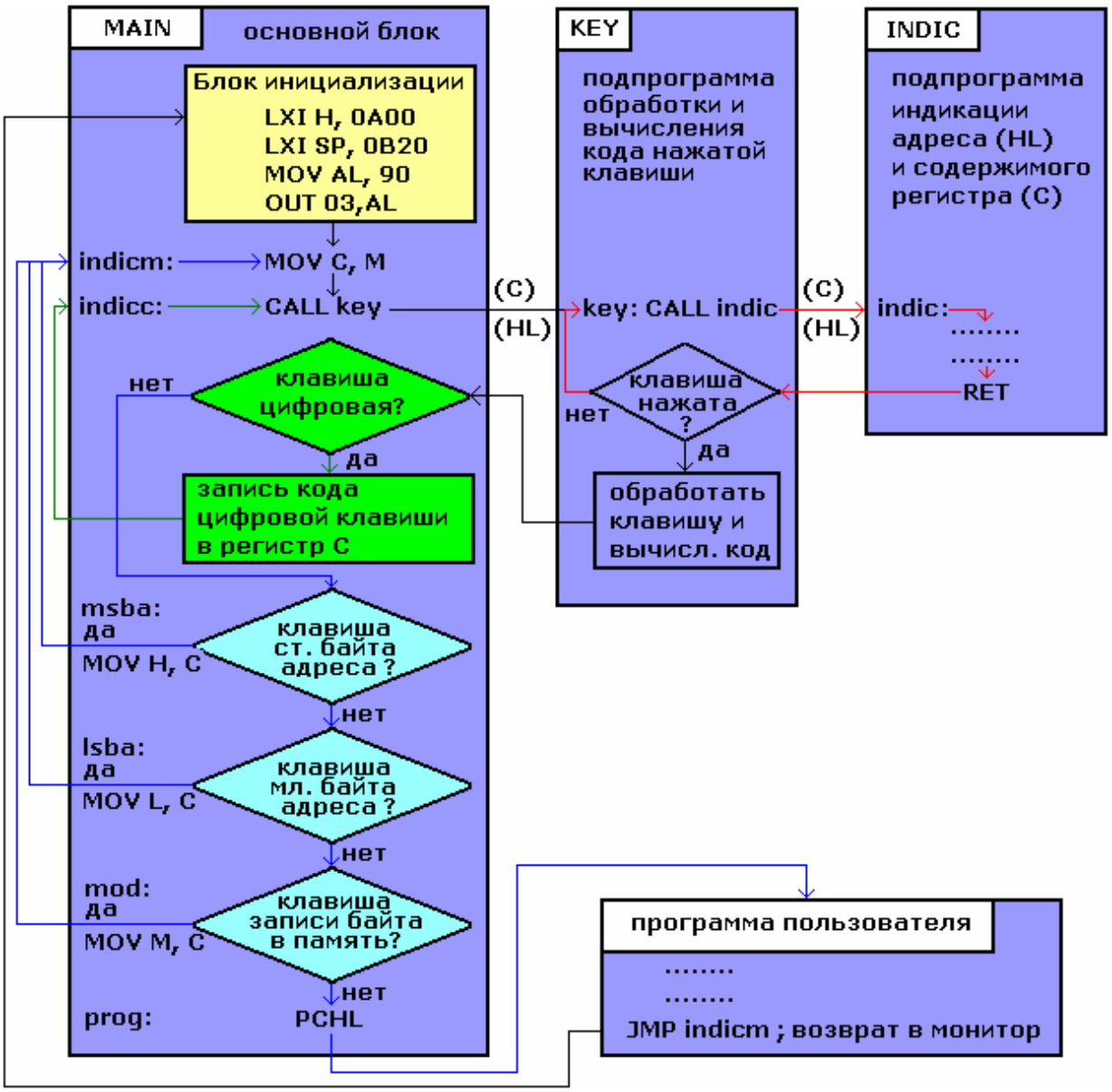


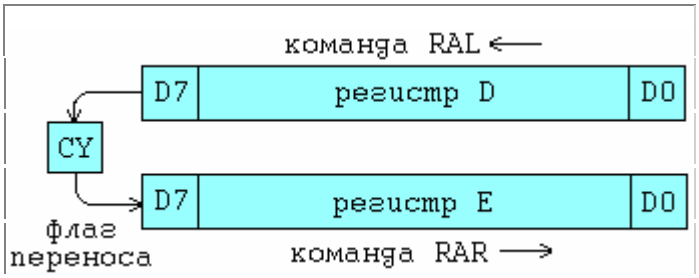
Рис.4.4 Блок-схема программы Монитор

метка	адрес	код	команда	комментарий и/или описание команды
#### основной блок монитора ####				
main	0800	26 00 0a	lxi H, 0a00	начальный адрес ОЗУ пользователя (коды программ и данных) далее может быть изменен
	0803	31 20 0b	lxi SP, 0b20	начальный указатель области стека ("дно")
	0806	3e 90	mvi A, 90	управляющий байт настройки портов ППИ: PB и PC на вывод, PA на ввод
	0808	d3 03	out 03	вывести управляющий байт в регистр управления ППИ

	080a	00	nop	холостая команда (для совместимости с другой версией)
indicm	080b	4e	mov C, M	переслать рабочий байт в буферный регистр C из ячейки памяти с адресом в паре регистров HL
indicc	080c	cd 51 08	call key	вызов подпрограммы вычисления кода нажатой клавиши
	080f	fe 04	cpi 04	нажата цифровая клавиша ?
	0811	d2 22 08	jnc msba	если нажата директивная клавиша, то перейти к метке msba
	0814	47	mov B, A	если нажата цифровая клавиша, то сохранить ее тетракод в регистре B
	0815	79	mov A, C	загрузить рабочий байт в аккумулятор
	0816	17	ral	сдвинуть влево рабочий байт в аккумуляторе на два бита,
	0817	17	ral	
	0818	00 00 00	nop nop nop	3 холостых команды (для совместимости с другой версией)
	081b	e6 fc	ani fc	освободить (обнулить) два крайних левых бита в аккумуляторе под тетракод нажатой клавиши (маска fc = 1111 1100)
	081d	b0	ora B	и записать тетракод из регистра B в два младших бита аккумулятора
	081e	4f	mov C, A	сохранить полученный рабочий байт в регистре C
	081f	c3 0c 08	jmp indicc	и высветить его (а также адрес) на линейках светодиодов
msba	0822	fe 04	cpi 04	нажата клавиша старшего байта адреса ?
	0824	c2 2b 08	jnz lsba	если нет, то перейти к метке lsba
	0827	61	mov H, C	если да, то поместить старший байт адреса в регистр H
	0828	c3 0b 08	jmp indicm	высветить новый адрес и его содержимое
lsba	082b	fe 05	cpi 05	нажата клавиша младшего байта адреса ?
	082d	c2 35 08	jnz mod	если нет, то перейти к метке mod
	0830	00	nop	
	0831	69	mov L, C	если да, то поместить старший байт

				адреса в регистр L
	0832	c3 0b 08	jmp indicm	высветить новый адрес и его содержимое
mod	0835	fe 06	cpi 06	нажата клавиша модификации данных ?
	0837	c2 40 08	jnz prog	если нет, то перейти к метке prog
	083a	00	nop	
	083b	71	mov M, C	если да, то поместить содержимое регистра C в ячейку памяти по указанному адресу
	083c	23	inx H	и перейти к следующему адресу
	083d	c3 0b 08	jmp indicM	высветить новый адрес и его содержимое
prog	0840	e9	pchl	(PC) <-- (HL) ; запуск программы пользователя
#### подпрограмма ожидания нажатия на клавишу и вычисления кода ####				
key				
nokey	0851	cd a1 08	call indic	подпрограмма индикации (HL) - адрес и данных (C)
	0854	db 00	in 00	опросить порт клавиатуры PA
	0856	fe ff	cpi ff	клавиша нажата ?
	0858	ca 51 08	jz nokey	если нет - ждать нажатия
	085b	5f	mov E, A	если да - временно сохранить сканкод в регистре E
	085c	cd 95 08	call delay	задержка ~ 10 мсек на время дребезга контактов клавиши при нажатии
waitoff	085f	db 00	in 00	снова опросить порт клавиатуры
	0861	fe ff	cpi ff	клавиша отпущена ?
	0863	c2 5f 08	jnz waitoff	если нет подождать
	0866	cd 95 08	call delay	задержка ~ 10 мсек на время дребезга контактов клавиши при отпускании
	0869	7b	mov A, E	вернуть сканкод в аккумулятор
shift:	086a	1e ff	mvi E, ff	в этом блоке команд производится преобразование сканкода в вычисленный
	086c	1f	rar	

	086d	1c	inr E	код путем сдвига сканкода вправо до того момента, когда единственный нулевой бит сканкода будет "вытолкнут" командой rar во флаг переноса. В этот момент в регистре E зафиксируется вычисленный код клавиши. Например нажата SW2. Ее сканкод = 1111 1011. Потребуется 3 сдвига, чтобы вытолкнуть "0", одновременно содержимое регистра E увеличится на 3 и станет равным ff +3 = 2.
	086e	da 6c 08	jc shift	
	0871	7b	mov A, E	
	0872	c9	ret	выйти из подпрограммы key
#### подпрограмма delay временной задержки на 10 мсек ####				
delay:	0895	d5	push D	сохранить значение рег-ра E (а не D!), т.к. он используется в п/п "key"
	0896	11 50 08	lxi D, 0850	записать в пару DE начальное значение временной задержки
dl:	0899	1b	dcx D	уменьшать в цикле величину задержки
	089a	7a	mov A, D	вычислить содержимое DE с помощью операции (D "ИЛИ" E) - нулевой результат только в том случае, если и (D) и (E) равны 0
	089b	b3	ora E	
	089c	c2 99 08	jnz dl	повторить, если результа не равен 0
	089f	d1	pop D	восстановить прежнее значение регистра E, по истечении задержки
	08a0	c9	ret	и выйти из подпрограммы
#### подпрограмма индикации (вывод адреса и байта данных на три линейки ####				
indic:	08a1	3e fb	mvi A, fb	записать в аккумулятор код линейки старшего байта адреса
	08a3	d3 01	out 01	и вывести его в порт PB с адресом 01
	08a5	7c	mov A, H	переслать в аккумулятор старший байт адреса
	08a6	cd ba 08	call lght	и высветить его
	08a9	3e f7	mvi A, f7	записать в аккумулятор код линейки младшего байта адреса
	08ab	d3 01	out 01	и вывести его в порт PB с адресом 01
	08ad	7d	mov A,L	переслать в аккумулятор младший байт

				адреса
	08ae	cd ba 08	call lght	и высветить его
	08b1	3e fe	mvi A, fe	высветить содержимое ячейки памяти по указанному адресу
	08b3	d3 01	out 01	
	08b5	79	mov A, C	
	08b6	cd ba 08	call lght	
	08b9	c9	ret	и выйти из подпрограммы
#### подпрограмма lght ####				
lght:	08ba	cd c8 08	call mirror	переставить биты в обратном порядке
	08bd	2f	cma	и проинвертировать их
	08be	d3 02	out 02	подключить аноды светодиодов
	08c0	cd db 08	call del	задержка для увеличения видимой яркости
	08c3	3e ff	mvi A, ff	погасить светодиоды
	08c5	d3 02	out 02	
	08c7	c9	ret	и выйти из подпрограммы
#### подпрограмма перестановки битов Mirror ####				
mirror:	08c8	c5	push B	сохранить значения регистров
	08c9	d5	push D	
	08ca	0e 08	mvi C, 08	регистр C - счетчик битов
	08cb	57	mov D, A	переслать в D биты, которые необходимо переставить
shift:	08cd	7a	mov A, D	
	08ce	17	ral	
	08cf	57	mov D, A	
	08d0	7b	mov A, E	
	08d1	1f	rar	
	08d2	5f	mov E, A	из диаграммы видно, что старший бит источника D7 станет младшим битом D0 приемника, аналогично будут переставлены местами остальные биты
	08d3	0d	dcr C	
	08d4	c2 cd 08	jnz shift	
	08d7	7b	mov A, E	вернуть в аккумулятор переставленные биты

	08d8	d1	pop D	восстановить значения регистров D и B
	08d9	c1	pop B	
	08da	c9	ret	и выйти из подпрограммы
#### подпрограмма задержки del ####				
del:	08db	3e ff	mvi A, ff	без комментариев!
dl:	08dd	3d	dcr A	
	08de	c2 dd 08	jnz dl	
	08e1	ret		и выйти из подпрограммы

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Получите вариант выполнения работы у преподавателя.
2. Введите изменения в некоторые численные значения программы в соответствие с заданием.
3. Потребуйте у преподавателя подключить плату расширения и включить лабораторный стенд.
4. Введите из листинга в УМК коды программы.
5. Запустите программу (при необходимости исправьте ошибки). Предъявите результат преподавателю.

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ №4

0. Назначение, состав и принцип действия PCI**.**
1. Объяснить работу схемы подключения USART к МП-системе.
2. Типовая последовательность битов в передаваемом кадре для асинхронного режима работы.
3. Расчет адресов портов ввода/вывода.
4. Назначение отдельных битов инструкции режима, команды управления и байта состояния УСАПП.
5. Уметь комментировать каждую команду программы передачи

ЛАБОРАТОРНАЯ РАБОТА №5
ПРОГРАММИРУЕМЫЙ СВЯЗНОЙ ИНТЕРФЕЙС (PCI)

ЗАДАНИЕ: Рассчитать параметры схемы и написать программу вывода.

Теоретические и справочные сведения.

Программируемый связной интерфейс или универсальный (синхронно-) асинхронный приемо-передатчик (У(С)АПП или U(S)ART) предназначен для организации обмена данными между МП и удаленными ВУ в последовательном формате. По этой причине УСАПП называют также последовательным интерфейсом (IOS).В качестве передатчика УСАПП преобразует параллельный код в последовательный и отправляет его в линию связи, а в качестве приемника осуществляет обратное преобразование. УСАПП может обмениваться данными с удаленными устройствами в симплексном (движение информации **в одном направлении**), полудуплексном (информация передается и принимается **в обоих направлениях, но поочередно**) и дуплексном режимах (обмен данными в **обоих направлениях одновременно**).

На рисунке 5.1 приведено упрощенное условное обозначение УСАПП, схема его включения в микропроцессорную систему и типичная последовательность бит на входе приемника или выходе передатчика в асинхронном режиме работы. Микропроцессор на схеме не показан. Счетчик СТ0 таймера (мог быть и другой) обеспечивает требуемую скорость обмена данными.

Назначение некоторых выводов: TxD - выход передатчика, RxD - вход приемника, CLK - вход частоты синхронизации, TxC - вход синхросигнала передатчика, RxC - вход синхросигнала приемника, ~CTS - инверсный вход готовности приемника терминала (удаленного устройства, в том числе такого же приемопередатчика или модема).

В простых системах связи вход ~CTS можно жестко связать с "землей", уведомляя локальный передатчик, что удаленный приемник "всегда готов", а его действительная готовность к приему - это забота программиста или оператора! Если используется стандартный протокол связи, например RS-232C, то вход ~CTS должен быть отсоединен от нулевого провода. C/~D - функциональный вход "управление/данные". Если C/~D = 0 , то МП и УСАПП обмениваются байтом данных, если C/~D = 1, то происходит запись байта управления или чтение байта состояния.

Назначение остальных выводов PCI аналогично назначению соответствующих выводов PPI. Приведенных на рисунке 5.1 выводов достаточно для реализации связи с не очень удаленными объектами, например с компьютером или с другими МП.

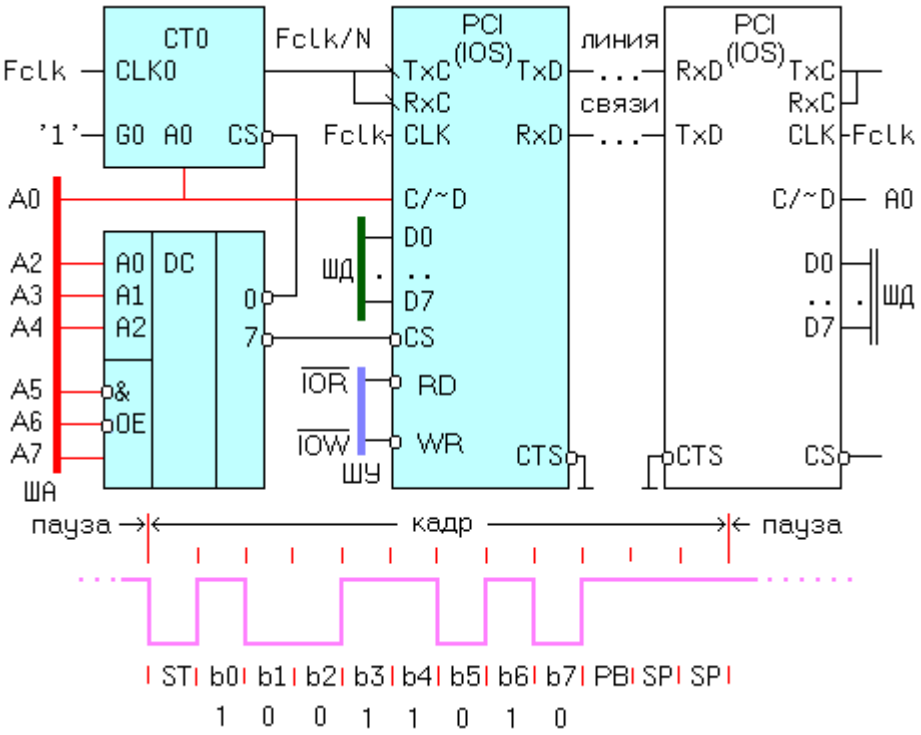


Рис. 5.1 Схема подключения PCI

Из приведенного рисунка нетрудно вычислить адреса PCI. Для нулевого выхода дешифратора, подключенного к входу таймеру "выбор микросхемы"адреса уже найдены. Активизация инверсного входа ~CS УСАПП производится подачей сигналов A4,A3,A2 = 111(BIN) = 7(DEC) и разрешающих работу дешифратора сигналов A7,A6,A5 = 100(BIN). В таблице приведены два из четырех возможных адресов PCI (линия A1 в схеме не используется, поэтому можно выбрать любое значение, например A1= 0).

ЛИНИИ ШИНЫ АДРЕСА								ДАННЫЕ / РЕГИСТР УПРАВЛЕНИЯ (CSR)	АДРЕС (HEX)
A7	A6	A5	A4	A3	A2	A1	A0		
1	0	0	1	1	1	x	0	Адрес порта данных	9C
1	0	0	1	1	1	x	1	Адрес порта CSR	9D

Одним из наиболее распространенных режимов работы УСАПП является асинхронный режим. В этом режиме каждый передаваемый (принимаемый) символ (кадр) содержит следующие поля:

- обязательный стартовый бит, на рисунке обозначен ST, всегда равен нулю,
- 5..8 информационных бит,
- необязательный бит контроля четности/нечетности PB,
- 1..2 стоп-бита SP.

Кадры следуют непрерывно или отделяются паузами. Информационные биты передаются начиная с младших разрядов. На рис.5.1 передается/принимается код **01011001**, а не 10011010.

УСАПП программируется записью в него байта управления, который может быть двух типов:

- **инструкция режима,**
- **команда управления.**
-

ФОРМАТ ИНСТРУКЦИИ РЕЖИМА

Инструкция режима задает режим синхронизации, формат данных, скорость обмена, необходимость контроля. Ниже в таблице 5.1 приведен формат инструкции режима.

D7	D6	D5	D4	D3	D2	D1	D0

Таблица 5.1

D7	D6	Число стоп-бит	Вид контроля	D5	D4
0	0	Запрет	Нет контроля	0	0
0	1	1 стоп-бит	Контроль нечетности	0	1
1	0	"полтора" стоп-бита	Нет контроля	1	0
1	1	2 стоп-бита	Контроль четности	1	1

D3	D2	Число информационных бит	Частота синхронизации	D1	D0
0	0	5	$f_{TxC(RxC)}$ / 1	0	1
0	1	6	$f_{TxC(RxC)}$ / 16	1	0
1	0	7	$f_{TxC(RxC)}$ / 64	1	1
1	1	8			

- Биты D7,D6 определяют число стоп-бит в каждом кадре, причем "полтора бита" обозначают длительность в полтора тактовых интервала. Обычно для экономии времени используется один стоп-бит.
- Длительность тактового интервала $(f_{TxC(RxC)}/x)^{-1}$сек задают биты D1,D0. Частота f_{TxC} или f_{RxC} - частота сигнала на одноименных входах TxC и RxC приемопередатчика должна быть по справочнику меньше или равной $f_{CLK}/(4.5)$. Внутри УСАПП может быть дополнительно поделена в 16 или 64 раза. Частота $f_{TxC(RxC)}/x$ определяет скорость передачи в Бодах (baud) или битах в секунду (bps). **Для УСАПП обе скорости совпадают. В асинхронном режиме комбинация D1,D0 = 01 недопустима.** Максимальную скорость обмена данными нетрудно вычислить. Пусть f_{CLKMax} = 2МГц, тогда $f_{TxC(RxC)}$ = $f_{CLK}/4.5$

=444444,44Гц. В асинхронном режиме дополнительный коэффициент деления должен быть не менее 16-ти (см. инструкцию режима). Поэтому максимальная скорость обмена равна 444444,44Гц / 16 = 27777 бит в секунду.

- Число информационных бит в кадре определяется битами D3,D2 инструкции режима.
- Биты D5,D4 задают вид контроля правильности передачи. Если производится контроль четности, то УСАПП-передатчик вычисляет "сумму по модулю два" информационных бит и полученный бит суммы вставляет в передаваемый кадр в качестве бита контроля четности. УСАПП-приемник снова вычисляет "сумму по модулю два" информационных бит и если значение принятого и вычисленного бита контроля совпадают - делается вывод об отсутствии искажений при передаче по линии связи. Этот метод позволяет выявлять только нечетное число искажений бит информации и самого бита контроля, поэтому обычно контроль четности/нечетности не применяется и длина кадра уменьшается на один тактовый интервал. В современных устройствах связи применяются более сложные методы контроля правильности передачи (подсчет контрольной суммы, контроль с помощью циклического избыточного кода - CRC , передача с подтверждением приема и др.).
-

ФОРМАТ КОМАНДЫ УПРАВЛЕНИЯ

Ниже в таблице 5.2 приведены некоторые биты команды управления.

Таблица 5.2

Разряд	Обозначение	НАЗНАЧЕНИЕ КОМАНДЫ
D0	TxEN	Разрешение работы УСАПП в качестве передатчика, D0=1
D2	RxEN	Разрешение работы УСАПП в качестве приемника, D2=1
D4	ER	Сброс в "0" флагов ошибок, D4=1
D6	RESET	Программный сброс УСАПП в исходное состояние, D6=1

ФОРМАТ БАЙТА СОСТОЯНИЯ

В процессе работы можно осуществлять контроль работы УСАПП путем чтения байта его состояния. Ниже в таблице 5.3 приведены некоторые наиболее употребительные в асинхронном режиме биты состояния УСАПП.

Таблица 5.3

D7	D6	D5	D4	D3	D2	D1	D0
		FE	OE	PE		**RxRDY**	**TxRDY**

Биты D5,D4,D3, называемые также флагами, устанавливаются (сбрасываются) приемником УСАПП сигнализируя:

- PE=1 (Parity Error), если УСАПП зафиксировал ошибку при контроле четности/нечетности,
- OE=1(Overrun Error), если была попытка считать в микропроцессор передаваемый из линии в приемник код, до завершения его полной передачи,
- FE=1(FraMe Error), если приемник не обнаружил стоп-бит(ы). Эта ошибка обычно возникает, если частота fTxC передатчика больше чем на 5% отличается от частоты fRxC приемника. Другой причиной м.б. неравенство информационных битов передатчика и приемника.

Бит RxRDY (Готовность приемника), если RxRDY=0, то приемник еще не преобразовал последовательный код в параллельный и считывать его в микропроцессор бессмысленно.

Бит TxRDY (Готовность передатчика) - если TxRDY=0, то передатчик еще не преобразовал параллельный код в последовательный и загрузка следующего кода из МП в передатчик исказит текущее передаваемое значение.

Ниже приводится фрагмент программы передачи массива байтов, хранящихся в памяти. Начальный адрес массива символов - 8007(HEX). Передается 8 байтов: fd,87,9c,88,40,55,8a,b8. Адреса таймера 80,83 и УСАПП 9c,9d были получены ранее. Коэффициент деления частоты f_{CLK}(N0=5)>4.5. Программа написана на ассемблере для МП семейства 80x86.

АДРЕС	КОД
8007	fd
8008	87
8009	9c
800a	88
800b	40
800c	55
800d	8a
800e	b8

```
;#### настройка и запуск делителя частоты
    mov al,1eh; CT0,1 мл.байт, режим 3, код BIN
    out 83h,al; вывод байта упр. 1e в порт 83
    mov al,5  ; коэфф. деления частоты Fclk N0=5
    out 80h,al; и запись его в CT0 по адресу 80

;#### настройка УСАПП
    ..........
    mov al,0; эти 4 команды - недокументированная
    out 9dh,al; особенность УСАПП. После 3х команд
```

```
    out 9dh,al; OUT ..... приемопередатчик гарантированно
    out 9dh,al; ожидает команду управления, тут-то и подаем программный сброс!

    mov al,40h;программный сброс D6=1
    out 9dh,al;

    mov al,0cfh;инструкция режима: 2 стоп-бита,
    out 9dh,al ;нет контроля,8 инф. бит, fTxC(RxC) / 64

    mov al,01h;разрешение передачи TxEN=1
    out 9dh,al;

;#### передача массива символов
st:  mov dl, 8; число передаваемых символов
    mov bx,8007h; начальный адрес блока
nxt: mov al,[bx]; переслать текущий байт в AL
    out 9ch,al; и вывести его в УСАПП (в этот момент и начинается преобр. кода)
wt:  in al,9dh; чтение байта состояния
    test al,1; символ передан в линию? (TxRDY=1?), 1-маска для выделения TxRDY
    jz wt; если нет,то подождать,
    inc bx; если да, перейти к адресу следующего байта
    dec dl; переданы все байты?
    jnz nxt; если нет,то повторить вывод след. байта
    jmp st; если да,то выйти из цикла (и начать сначала). Эта команда позволяет получить периодическую осциллограмму, которую удобно наблюдать.
```

Ниже приведены аналогичные команды ассемблера для УМК (в другом порядке). Используются регистры: a, d и пара bc.

```
out 9d ; код двухбайтовой команды = D3
in 9d ; код двухбайтовой команды = DB
Mvi a, 1 ; код двухбайтовой команды = 3E
ldax b ; код однобайтовой команды = 0A
Mvi d, 5 ; код двухбайтовой команды = 16
jz Metka ; код трехбайтовой команды = CA
dcr d ; код однобайтовой команды = 15
inx b ; код однобайтовой команды = 03
ani 1 ; код двухбайтовой команды = E6
```

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1. Получите вариант выполнения работы у преподавателя (физический адрес - "прог", адрес массива символов и их число, временную диаграмму и номера двух выходов дешифратора).
2. Перепишите программу с ассемблера для МП 80х86 на ассемблер для УМК.
3. Введите изменения в некоторые численные значения программы в соответствие с заданием и заполните листинг.
4. Потребуйте у преподавателя подключить плату расширения с УСАПП и включить лабораторный стенд.
5. Введите из листинга в УМК коды программы.
6. Запустите программу (при необходимости исправьте ошибки). На экране осциллографа вы должны увидеть заданную временную диаграмму передаваемого кадра. Предъявите ее преподавателю.

ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ №5

0. Назначение, состав и принцип действия PCI**.**
1. Объяснить работу схемы подключения USART к МП-системе.
2. Типовая последовательность битов в передаваемом кадре для асинхронного режима работы.
3. Расчет адресов портов ввода/вывода.
4. Назначение отдельных битов инструкции режима, команды управления и байта состояния УСАПП.
5. Уметь комментировать каждую команду программы передачи

ЛАБОРАТОРНАЯ РАБОТА №20
РАЗРАБОТКА ГЕНЕРАТОРА ИМПУЛЬСНОЙ ПОСЛЕДОВАТЕЛЬНОСТИ ЗАДАННОЙ ФОРМЫ

Цель работы: Изучение синтеза комбинационных схем.

Этапы проектирования.

1. Заданную последовательность разбиваем на ряд одинаковых интервалов N, на каждом из которых сигнал сохраняет постоянное значение.

2. Находим число независимых переменных: $n>=\log_2 N$ (ближайшее большее целое число).

3. В качестве примера синтезируем импульсную последовательность Y, приведенную на рисунке 20.1 (высокий уровень Y соответствует логической "1", а низкий - "0").

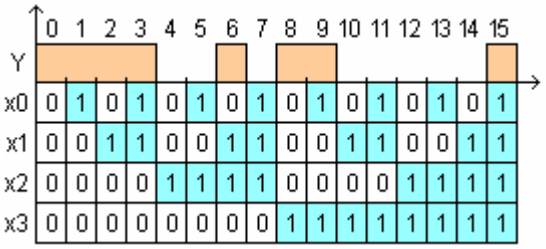


Рис. 20.1 Таблица истинности (временная диаграмма)

Приведенная последовательность (логическая функция Y), как видно из рисунка, разбивается на N=16 интервалов от 0 до 15-ти, причем число ее аргументов равно $n=\log_2 16=4$. Такое табличное задание логической функции (ЛФ) в зависимости от значений входных переменных называется таблицей истинности (ТИ).

4. Найдем алгебраическое выражение заданной ЛФ и заодно минимизируем ее с помощью таблицы Карно (ТК). Таблица Карно является разновидностью таблицы истинности (рис. 20.2).

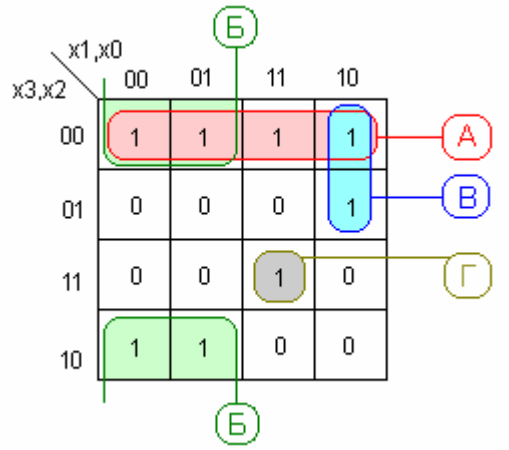


Рис. 20.2 Таблица Карно

4.1. Слева и сверху от ТК располагаются значения переменных x3,x2 и x1,x0, причем соседние клетки ТК отличаются значением ТОЛЬКО одного аргумента. Отсюда следует, что противоположные клетки в ТК также являются соседними и таблицу сначала можно "склеить" в цилиндр, а затем с помощью еще одной "склейки" превратить в непрерывную тороидальную поверность ("сушку", как выразился один студент).

4.2. В клетки ТК переносим соответствующие значения ЛФ из таблицы истинности. Например $Y_0=1$ при x3x2x1x0=0000 из крайнего левого (нулевого) столбца таблицы истинности записываем в левую верхнюю клетку ТК, т.к. этой клетке соответствуют те же аргументы x3x2x1x0=0000. Значение $Y_4=0$ при x3x2x1x0=0100 из 4-го столбца переносим в клетку ТК с "координатами" x3x2=01 и x1x0=00 и так далее для всех остальных значений ЛФ.

ВНИМАНИЕ! В столбцах 0,1,2,3 таблицы истинности (рис.20.2)

аргументы x1,x0 следуют в порядке 00,01,10,11, а в столбцах ТК в порядке 00,01,**11,10**. Аналогично обстоит дело и с остальными столбцами ТИ. Например Y_4=0,Y_5=0,Y_6=1,Y_7=0 записывается во вторую сверху строчку ТК в порядке 0001(Y_4=0,Y_5=0,**Y_7=0,Y_6=1**). Можно сказать, что 2 правых столбца ТК переставлены местами. Точно также "переставлены местами" две нижних строчки ТК.

4.3. СОСЕДНИЕ клетки ТК с единичными значениями ЛФ объединяем в прямоугольники (импликанты). В прямоугольнике может быть ТОЛЬКО 2^k (т.е.1,2,4,8...) клеток. Чем больше клеток в прямоугольнике и чем меньше прямоугольников, тем проще результирующее алгебраическое выражение ЛФ. Прямоугольники могут перекрываться.

4.4. Для каждого прямоугольника записывается логическое произведение (логическое И) тех переменных, которые в соседних клетках не изменяют своего значения. Причем, если значение переменной в соседних клетках ТК равно единице, то переменная записывается в прямом виде. Если значение переменной равно нулю, то в - инверсном.

В нашем примере оптимальным решением является объединение 8-ми единичных клеток в 4-ре прямоугольника (А,Б,В и Г), как показано на рисунке. Например для прямоугольника "В" переменные x1,x0 входят в обе клетки без изменения, поэтому в логическом произведении они обе будут присутствовать - x1 в прямом виде (x1=1), а x0 в - инверсном (x0=0). В прямоугольнике "В" присутствуют также переменные x3,x2, но x2 в соседних клетках принимает различные значения, поэтому в произведение войдет только переменная x3 в инверсном виде, т.к. x3=0. Таким образом для прямоугольника "В" логическое произведение: В = ~x3*x1*~x0, где "~" обозначает инверсию.

Для прямоугольника "А" (фрагмент ТК приведен внизу) во всех парах соседних клеток обе переменные x1 и x0 изменяют значение, поэтому в произведение x1 и x0 не войдут. Наоборот, переменные x3,x2 во все 4-ре клетки входят без изменения и равны при этом нулю. Следовательно логическое произведение для прямоугольника "А" будет иметь вид: А = ~x3*~x2. На рисунке 20.2.1соседние пары, меняющие значения выделены скобками.

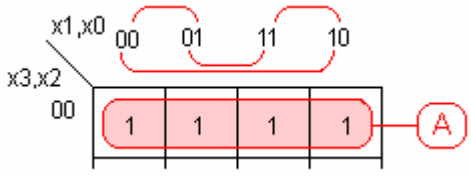


Рис. 20.2.1 Пример объединения клеток в ТК

В прмоугольнике "Б" в соседних колонках не меняет своего значения переменная x1, а в обоих строчках этого прямоугольника постоянна переменная x2. Учитывая, что значения обеих переменных равны нулю, в произведение они войдут в инверсном виде: Б = ~x2*~x1.

Последний прямоугольник "Г" состоит из одной клетки, т.к. "единичных соседей" у него нет: Г = x3*x2*x1*x0 (все переменные в прямом виде, т.к. их значения равны 1).

4.5. Полученные логические произведения объединяются с помощью логического ИЛИ (или логического сложения) в искомую ЛФ в базисе "И-ИЛИ-НЕ":

$$Y = А + Б + В + Г = \overline{x3}*\overline{x2} + \overline{x2}*\overline{x1} + \overline{x3}*x1*\overline{x0} + x3*x2*x1*x0 \quad (I)$$

Используя аксиому двойного отрицания и теорему двойственности получим выражение искомой ЛФ в базисе "И-НЕ":

$$Y = А + Б + В + Г = \overline{\overline{А + Б + В + Г}} = \overline{\overline{А} * \overline{Б} * \overline{В} * \overline{Г}} =$$
$$= \overline{\overline{\overline{x3}*\overline{x2}} * \overline{\overline{x2}*\overline{x1}} * \overline{\overline{x3}*x1*\overline{x0}} * \overline{x3*x2*x1*x0}} \quad (II)$$

4.6. Схемная реализация полученных алгебраических выражений. На схемах логическое "И" обозначается знаком "&", логическое "ИЛИ" - знаком "1" и логическое отрицание (логическое "НЕ" или инверсия) обозначается кружком.

В схеме (II) на рисунке 20.2.2 используются только элементы "И-НЕ". На рисунке приведены обозначения логических элементов "И", "ИЛИ", "И-НЕ" в соответствии с ГОСТ 2.743-91.

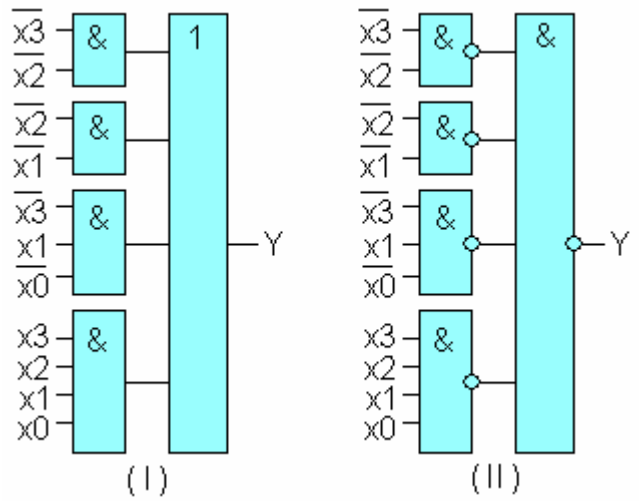


Рис. 20.2.2 Две реализации одной схемы

Сборка и отладка схемы.

5. Запустите интегрированную среду разработки и отладки электронных схем Electronic Workbench (EW) . В открывшемся рабочем пространстве (рис. 20.2.3) переименуйте проект с обезличенного Untitled в XXXX_lab20, где XXXX номер вашей группы (например 8888_lab20). Для этого кликните в рабочем меню п. File | Save As ... и в открывшемся диалоговом окне введите имя рабочего файла 8888_lab20.ewb, папку и настройки оставьте по умолчанию.

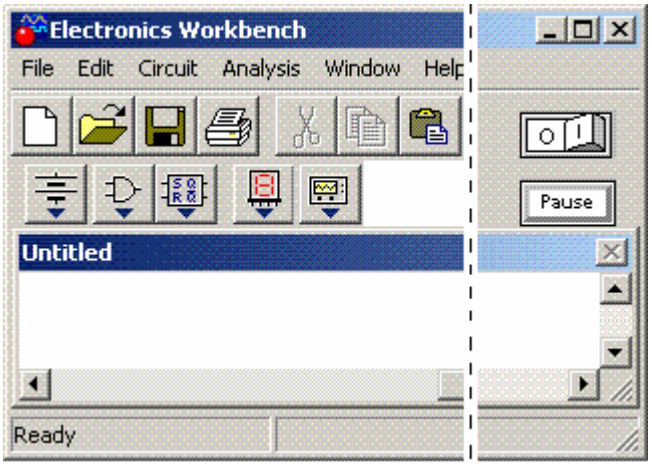


Рис. 20.2.3 Главное окно Electronic Workbench

5.1. Приступим к сборке схемы. Кликните по иконке логических элементов (ЛЭ) на панели инструментов EW. В открывшемся "ящике" нам понадобятся следующие ЛЭ. Слева по порядку: "И &", "ИЛИ 1", "НЕ", "ИЛИ-НЕ", "И-НЕ" в зарубежном обозначении (рис. 20.2.4).

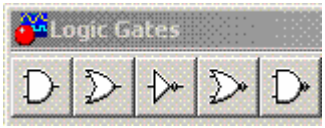


Рис. 20.2.4 Условные обозначения основных ЛЭ

Для реализации схемы (I) понадобятся ЛЭ "И", "ИЛИ" и "НЕ", а для схемы (II) - элементы "И_НЕ","НЕ". Кроме того понадобятся некоторые инструменты, находящиеся под иконкой . Это осциллограф и генератор слов, на рисунке 20.2.5 отмечены стрелками.

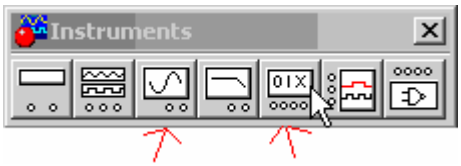


Рис. 20.2.5 Осциллограф и генератор слов

5.2. Для реализации схемы (I) понадобятся 4 логических элемента "И", один ЛЭ "ИЛИ" и 4 инвертора. Поместим их на рабочее поле нашего чертежа. Для этого подведите курсор к соответствующему ЛЭ, нажмите на левую кнопку и вытащите элемент на рабочее поле.

Из рисунка 20.2.6 видно, что все ЛЭ, кроме инверторов имеют два входа, а в схеме два двухвходовых ЛЭ "И", один трехвходовый и один четырехвходовый. Кроме того требуется четырехвходовый элемент "ИЛИ". Изменить число входов можно сделав "правый клик" на нужном ЛЭ и далее в п. контекстного меню CoMponent Properties... | Number of Inputs отметить нужное число. Развернуть ЛЭ на чертеже можно вызвав правым кликом

контекстное меню и затем использовать п. меню Rotate или Flip...

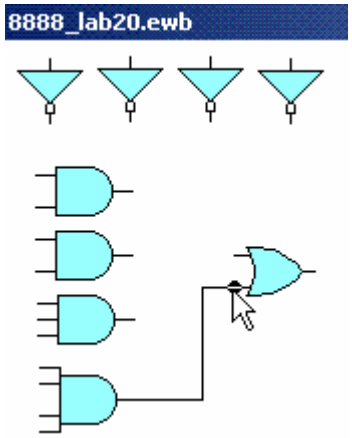


Рис. 20.2.6 Расположение ЛЭ на чертеже

Необходимые соединения делаются следующим образом: подведите курсор к окончанию входа или выхода, появится "электрический" курсор в виде жирной точки. Зажмите левую кнопку и тяните проводник к другому входу или выходу в соответствии со схемой до появления следующей точки, затем отпустите кнопку.

5.3. Не забудьте подключить к схеме осциллограф и генератор слов. Окончательно схема (I) будет выглядеть примерно таким образом (рис. 20.3):

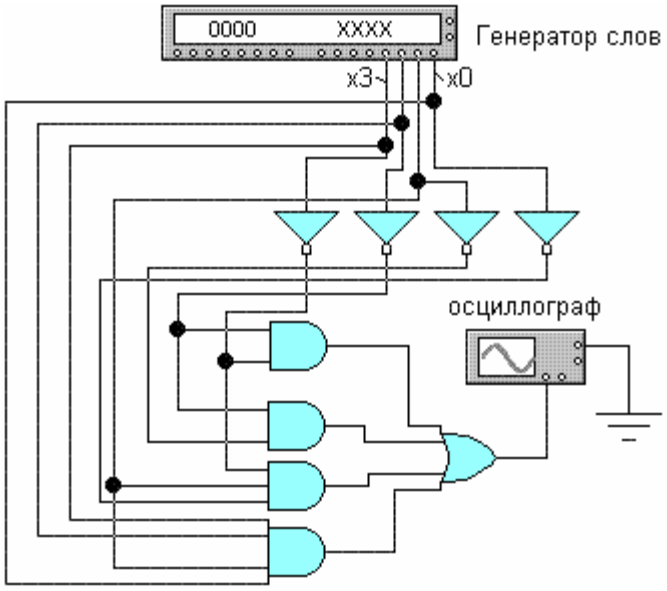


Рис. 20.3 Окончательная схема

Двойным кликом откройте рабочую панель генератора слов и измените следующие установки: конечное значение = F_{16} (15_{10}), затем кликнув по кнопке "Pattern" задайте режим работы генератора слов - "Up counter", что означает работу в режиме суммирующего счетчика (счетчики будут изучаться в курсе отдельно). Изменения отмечены на рисунке 20.3.1.

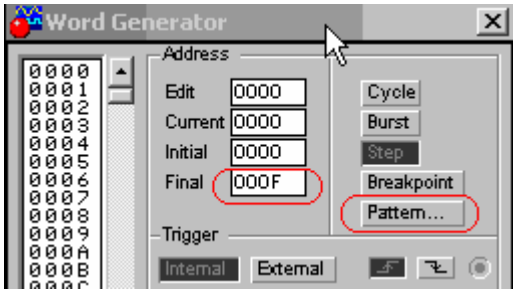


Рис. 20.3.1 Настройка генератора слов

Теперь откройте панель осциллографа, установите масштаб времени 2мсек/деление и включите установку переключателем  . Зафиксировать временную диаграмму на эране осциллографа лучше всего в пошаговом режиме "Step" генератора слов. Как и должно быть экспериментально полученная временная диаграмма (рис. 20.4) совпала с заданной временной последовательностью (рис. 20.1). На рисунке выделен один период повторения.

На экране осциллографа (рис. 20.4) виден также незапланированный короткий всплеск (импульсная помеха). Появление таких зловредных "пичков" обусловлено конечным значением времени задержки распространания сигналов при прохождении через элементы схемы $t_{зд.р}>0$ (или t_{pd} в зарубежном обозначении). В частности, триггеры генератора слов переключаются не одновременно, а последовательно начиная с младшего разряда. Таким образом, переход выходных значений генератора слов от комбинации x3x2x1x0 = 1111 к комбинации x3x2x1x0 = 0000 происходит не одновременно. Сначала переключается младший триггер x0 и появляется код 1110 (14-й интервал), затем переключается триггер x1 - появляется код 1100 (12-й интервал), далее x2 - код равен 1000 (8-й интервал) и наконец-то схема преходит к коду 0000. Таким образом при переполнении счетчика возникают три незапланированные короткие кодовые комбинации.

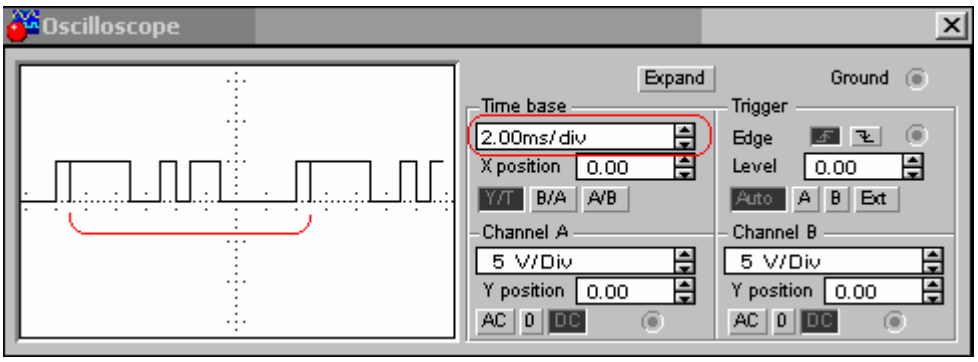


Рис. 20.4 Временная диаграмма

Проанализируем, как поведет себя разработанная нами схема при этих входных значениях. Для этого снова обратимся к рис. 20-1 с заданной импульсной последовательностью. При комбинации x3x2x1x0=1000 выходное значение Y=1, т.е. единичный уровень выходного сигнала

прерываться не будет. А вот при значениях x3x2x1x0=1100 и 1110 выходное значение Y=0, т.е. единичный уровень выходного сигнала сменится в течение короткого времени равного 2* $t_{зд.р}$, что и приводит к кратковременной помехе, которая видна на экране осциллографа. Естественно, такой же вывод можно получить подставляя значения x3x2x1x0=1100 и 1110 в формулу (I) или (II). Методы подавления таких помех изложены в соответствующих разделах учебного курса.

5.4. Реализация схемы (II) и проверка ее работоспособности проводятся аналогично.

Выполнение работы:

6. Получите задание, проведите соответствующий расчет, поочередно соберите схемы (I) и (II) и покажите временные диаграммы преподавателю.

ЛАБОРАТОРНАЯ РАБОТА №21
СИНТЕЗ ПРЕОБРАЗОВАТЕЛЯ ДВОИЧНОГО (ДВОИЧНО-ДЕСЯТИЧНОГО) КОДА В СЕМИСЕГМЕНТНЫЙ

Цель работы: Изучение преобразователей кодов.

На рисунке 21.1 показан фрагмент подключения одного светодиода (сегмента) к выходу преобразователя с открытыми коллекторами и приведены начертания цифр. В качестве примера рассмотрим разработку преобразователя двоично-десятичного кода в семисегментный (BCD/7seg).

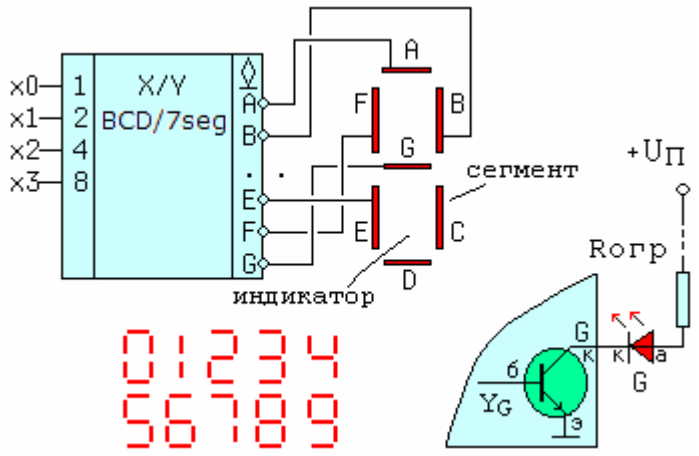


Рис.21.1 Блок схема преобразователя и фрагмент подключения отдельного светодиода (сегмента)

Этапы проектирования.

Такой преобразователь должен иметь четыре входа, т.к. для кодирования десятичных цифр от 0 до 9 достаточно $\log_2 10 = 4$-х двоичных, и семь выходов, по одному на каждый сегмент.

Сформулируем условия свечения/гашения светодиода:

1) Светодиод включен, если напряжение на его аноде больше, чем на катоде (о конкретных значениих напряжения и тока пока речь не идет). Анод через ограничивающий ток резистор уже подключен к плюсу источника питания, поэтому на катоде должен быть потенциал близкий к нулю. Для этого n-p-n транзистор, работающий в ключевом режиме, должен быть открыт. Тогда потенциал его коллектора близок к нулю. Транзистор открыт, если потенциал на его базе больше нуля, т.е. должно быть**YG =1** (YG - логическая переменная, соответствующая сегменту G).

2) Светодиод выключен, если потенциалы его анода и катода равны. Это достигается, если ключевой транзистор закрыт и через него не протекает ток. Потенциал базы в этом случае должен быть равен нулю, т.е. **YG =0**.

Теперь, в соответствии с полученными условиями, заполним таблицу истинности преобразователя. Например в цифре 0 должны светиться все сегменты за исключением сегмента G. В цифре 1 светятся только два сегмента B и C и т.д. Весовые коэффициенты b^i двоично-десятичных разрядов равны 2^i (8,4,2 и 1). На рис.21.2 приведена таблица истинности. В таблице заполнена только колонка для сегмента A. Нули в ней проставлены для тех цифр, в которых сегмент A не светится.

Десятич цифра	8 x3	4 x2	2 x1	1 x0	сегменты $Y_A Y_B Y_C Y_D Y_E Y_F Y_G$
0	0	0	0	0	1 · · ·
1	0	0	0	1	0 · ·
2	0	0	1	0	1 ·
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1

Рис.21.2 Таблица истинности для сегмента A

В общем случае для синтеза этого ПК требуется составить семь уравнений. Найдем одно, для сегмента A, заполнив для него таблицу Карно. Ниже на рисунке приведена ТК прямого значения функции Ya сегмента A. Когда "нулевых" клеток в таблице меньше, чем "единичных" и/или они компактно сгруппированы (рис.21.3), целесообразно искать алгебраическое выражение инверсной логической функции, т.е. ~Ya. Логическая функция при этом может получиться значительно проще, т.е. содержать меньше переменных и слагаемых.

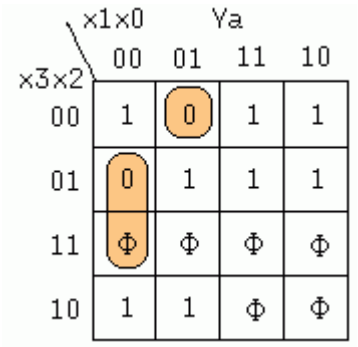


Рис. 21.3 Таблица Карно для сегмента А

Шесть значений ЛФ в таблице не определены (Ф) из-за отсутствия десятичных цифр больших девятки, поэтому для минимизации алгебраического выражения ЛФ доопределяем некоторые из них. Для нахождения инверсного значения ~Ya необходимо нарисовать еще одну ТК с инвертированнуми значениями ЛФ. Можно проделать это мысленно и объединить в прямоугольники клетки с нулевыми значениями ЛФ. Из таблицы найдем: ~Ya = x2*~x1*~x0 + ~x3*~x2*~x1*x0. Проинвертировав полученное выражение получим окончательный результат:

$$Ya = \overline{x2*\overline{x1}*\overline{x0} + \overline{x3}*\overline{x2}*\overline{x1}*x0}$$

В задании для работы будут варьироваться сегменты, количество входных двоичных наборов (10...16), а также начертания некоторых цифр.

Выполнение работы:

а) получите задание,

б) заполните таблицу истинности для заданного сегмента(ов) и соответствующую ему таблицу Карно,

в) вычислите алгебраическое уравнение,

г) запустите Electronic WorkBench (см. п.5 лабораторной работы №20) и соберите свою схему аналогично рис.20-3, только вместо осциллографа к выходам преобразователя подключите 7-ми сегментный индикатор,

д) покажите преподавателю свечение сегмента в соответствущих кодах на входах преобразователя.

ЛАБОРАТОРНАЯ РАБОТА №22
РАЗРАБОТКА СХЕМ НА РЕГИСТРАХ СДВИГА С ОБРАТНОЙ СВЯЗЬЮ

Цель работы: Изучение работы регистра сдвига.

Такие схемы используются в технике связи, для кодирования (декодирования) данных, при нахождении циклического избыточного кода - CRC (при архивировании, передаче данных, ...) и т.д.

На рисунке 22.1 приведено схематичное обозначение трехразрядного регистра сдвига и диаграммы его состояний. С приходом очередного импульса сдвига входное значение Foc записывается в младший разряд и одновременно содержимое регистра сдвигается на один разряд влево (регистр переходит в новое состояние). Составим диаграмму, начиная с нулевого состояния. Если в триггер младшего разряда Q0 записывается 0, то на диаграмме новое состояние записывается слева, а если - 1, то - справа. Каждому переходу соответствует импульс сдвига.

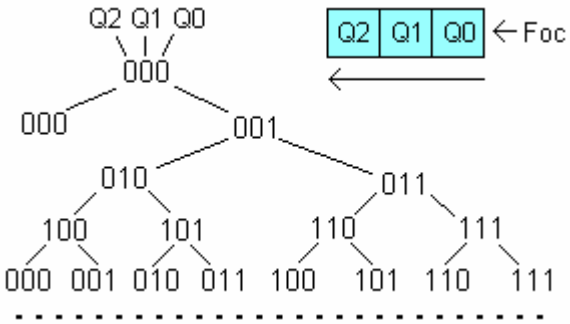


Рис. 22.1 Диаграмма состояний 3-х разрядного регистра сдвига

Объединяя одноименные узлы древовидной диаграммы, получим кольцевую диаграмму состояний трехразрядного регистра сдвига (рис. 22.2).

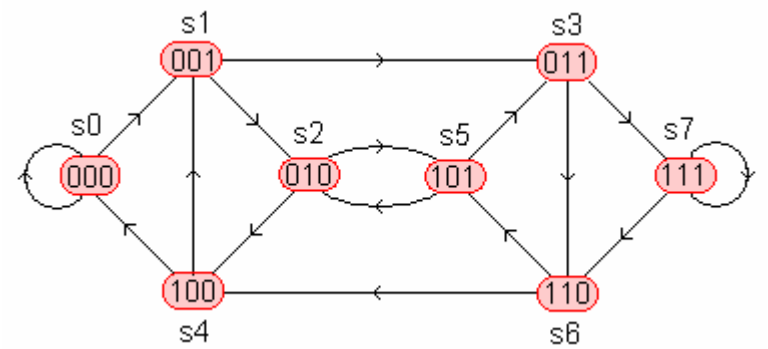


Рис. 22.2 Кольцевая диаграмма состояний 3-х разрядного регистра сдвига

Расчет схемы.

На рис. 22.3 приведен вариант типового устройства с комбинационной логической схемой (КЛС1) в цепи обратной связи. КЛС1 обеспечивает требуемые переходы. КЛС2 нужна только в том случае, если заданный алгоритм переходов не обеспечивается диаграммой состояний. Выходное значение Foc КЛС1 записывается в младший триггер регистра с приходом очередного активного фронта импульса сдвига.

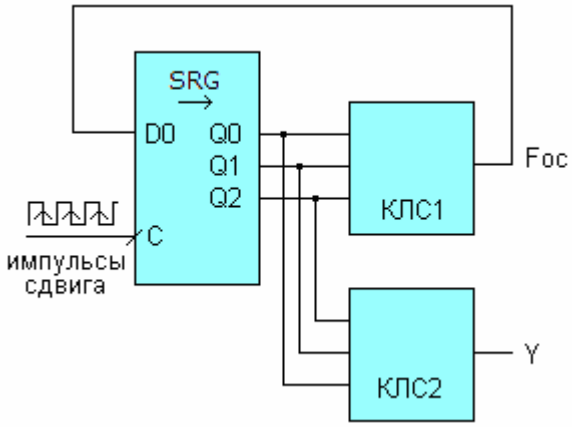
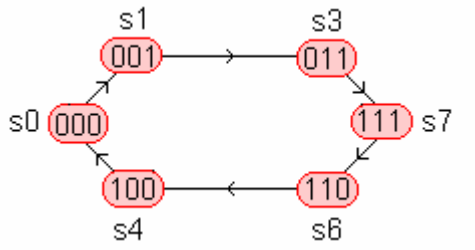


Рис. 22.3 Пример схемы

В качестве примера рассмотрим синтез схемы (техническое задание или ТЗ) имеющей 6 состояний: s0 -> s1 ->s3 -> s7 ->s6 -> s4 -> s0 .....



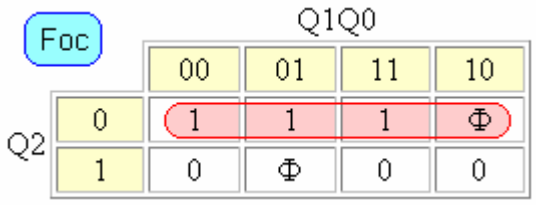
Составим таблицу состояний. КЛС1 должна генерировать при текущем состоянии si значение Foc, которое с приходом тактового импульса записывается в младший разряд регистра Q0, как показано в таблице 22.1.

Таблица 22.1

состояние	Q2 Q1 Q0	Foc
s0	000	1
s1	001	1
s3	011	1
s7	111	0
s6	110	0
s4	100	0
s0	000	1
....	....	....

Переходим к расчету логической функции Foc КЛС1, для этого заполняем таблицу Карно. Символом Ф, как всегда обозначаем неопределенное на этапе технического задания значение логической

функции (в ТЗ незадействованы и соответственно неопределены состояния 010 и 101).



Объединяя отмеченные в ТК значения, получаем "очевидное" решение:

Foc = ~Q2. (формула I)

Проверим, как поведет себя логическая функция Foc при двух неиспользуемых состояниях: Q2Q1Q0 = 010 и Q2Q1Q0 = 101. 1). При Q2Q1Q0 = 010 логическая функция Foc(010) = ~Q2 = 1 и с приходом очередного импульса сдвига регистр перейдет в состояние Q2Q1Q0 = 101. 2). При Q2Q1Q0 = 101 логическая функция Foc(101) = ~Q2 = 0 и с приходом очередного импульса сдвига регистр перейдет в состояние Q2Q1Q0 = 010 и далее процесс зациклится на этих двух состояниях.

Таким образом, проверка выявила побочный (незапланированный) бесконечный цикл, в который схема может перейти при включении питания, при воздействии помехи и т.д. Окончательный вид диаграммы состояний для нашего "очевидного" решения с учетом проведенного анализа будет выглядеть следующим образом (рис. 22.4):

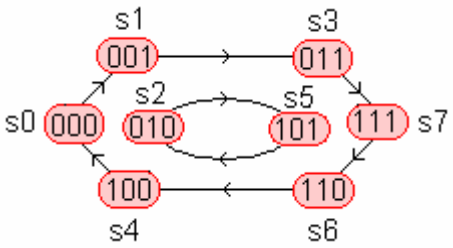


Рис. 22.4 Вариант диаграммы переходов с побочным циклом

Естественно, такое решение не может нас устроить. Выход из "порочного" цикла в нашем варианте можно осуществить тремя способами: переопределив одну или две связи в диаграмме состояний (на рис. 22.5 показаны схематично). В вариантах А и Б выход из побочного цикла гарантированно произойдет в течение двух тактовых интервалов импульсов сдвига, а в варианте В только за один, но в этом случае выражение для Foc будет сложнее (выбор "быстрого" или менее сложного решения, как всегда за разработчиком).

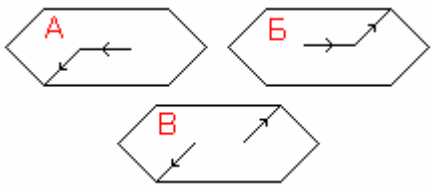
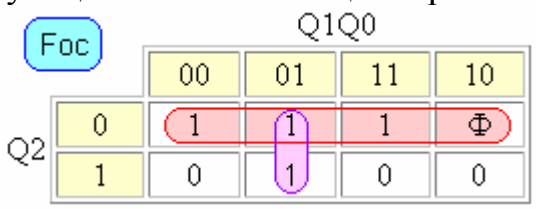


Рис. 22.5. Три варианта правильного решения задачи

Для корректировки технического решения выберем вариант Б. Как видно из рисунка, в этом случае необходимо переопределить переход из состояния 101 не к состоянию 010, а к 011. То есть, значение Foc(101) должно быть равно не нулю, а единице: Foc(101)=1. Подкорректируем значение в соответствующей клетке таблицы Карно:



Окончательно диаграмма состояний и алгебраическое решение будут выглядеть следующим образом (рис.22.6).

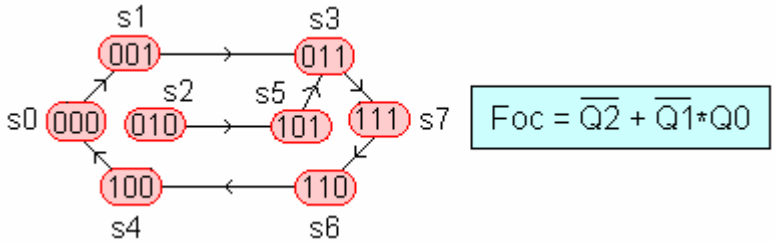


Рис. 22.6. Одно из правильных решений

Рисунок КЛС1, соответствующий формуле на рис. 22.6, не приводится.

Выполнение работы:

а) получите задание,

б) выявите возможные побочные циклы (если они есть) и выходы из них,

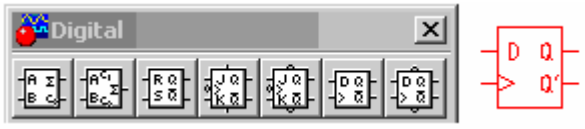
в) нарисуйте окончательную диаграмму состояний аналогичную рисунку 22.6 но, естественно, для вашего варианта,

г) заполните таблицу состояний и соответствующую ей таблицу Карно,

д) вычислите алгебраическое уравнение,

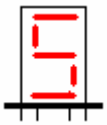
е) нарисуйте две схемы КЛС1: в базисе "И-ИЛИ-НЕ" и в базисе "И-НЕ" (см. лабораторную работу №20),

ж) запустите Electronic WorkBench (см. п.5 лабораторной работы №20) и соберите свою схему аналогично рис.22.3, причем схему регистра сдвига возьмите из конспекта, а D-триггер выберите следующим:

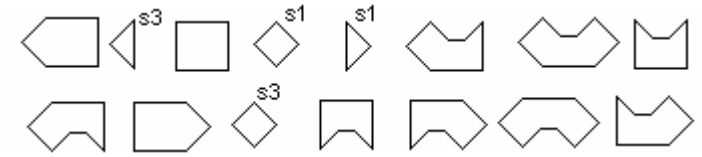


и) тактовые импульсы сдвига можно получить в пошаговом режиме работы генератора слов на крайнем правом его выходе (см. п. 5.3 лаб. работы №20),

к) для контроля состояний используйте 7-ми сегментный индикатор со встроенным преобразователем кода (см. лаб. работу №21).



л) смену состояний в соответствии с заданием предъявите преподавателю.
м) некоторые варианты заданий:



ЛАБОРАТОРНАЯ РАБОТА №23
СИНТЕЗ СЧЕТЧИКА С ПРОИЗВОЛЬНЫМ МОДУЛЕМ СЧЕТА $M<2^n$

Цель работы: Изучение построения и функционирования счетчиков.

Для работы выбрана одна из множества схем (рис.23.1), например, на основе 4-х разрядного асинхронного счетчика (свойства и функционирование счетчика описаны в конспекте лекций).

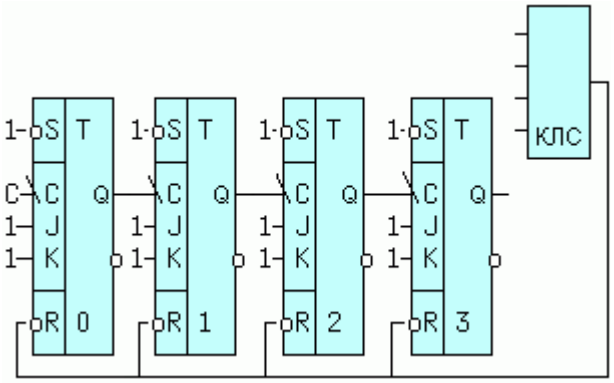


Рис.23.1 Четырехразрядный двоичный счетчик

Этапы проектирования.

В качестве примера проектирования выберем M=10 (двоично-десятичный счетчик, широко использующийся в схемотехнике). Такой счетчик должен иметь M=10 состояний от 0 до M-1 т.е. до 9-ти. С приходом каждого 10-го импульса счетчик должен возвращаться в исходное состояние, например в нулевое. Число триггеров находим из формулы: $n>=\log_2 M$ (ближайшее большее целое число), т.е. n=4.

Перевод в нулевое состояние всех триггеров можно осуществить, сформировав с помощью комбинационной логической схемы (КЛС) сигнал Yoc=~R=0 (~S=1), который "сбросит" все триггеры в момент поступления 10-го импульса. Во время счета от нуля до 9-ти сигнал Yoc должен принимать пассивное значение Yoc=1. В соответствии с этим описанием, заполним таблицу состояний двоично-десятичного счетчика.

Таблица 23.1

состояние	Q3Q2Q1Q0	Yoc
0	0000	1
1	0001	1
2	0010	1
3	0011	1
4	0100	1
5	0101	1
6	0110	1
7	0111	1
8	1000	1
9	1001	1
10	1010	0

Вычисление алгебраического выражения Yoc произведем с помощью таблицы Карно 23.2.

Таблица 23.2

Yoc	Q1Q0 = 00	01	11	10
Q3Q2 = 00	1	1	1	1
01	1	1	1	1
11	Ф	Ф	Ф	Ф
10	1	1	Ф	0

Пять состояний счетчика в таблице состояний 23.2 не определены и могут быть использованы для получения более компактной формулы (см. п.4.3 лаб. работы №20). Найдем инверсное значение ~Yoc = Q3*Q1 (см. лаб. работу №21). Окончательное выражение для Yoc будет иметь следующий вид:

$$Yoc = \overline{Q3 * Q1}$$

Теперь схема (рис. 23.2) примет законченный вид:

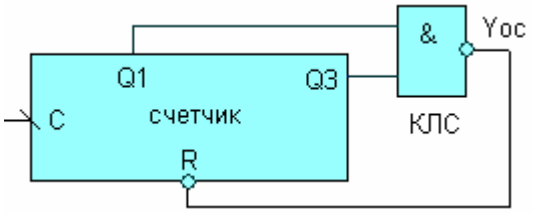
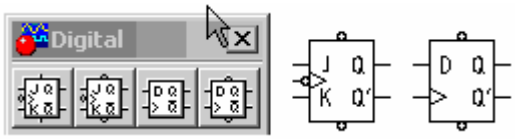


Рис.23.2 Схема счетчика по модулю 10

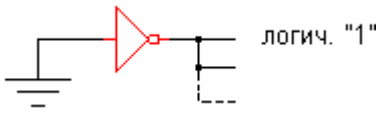
ВНИМАНИЕ: Варианты задания будут усложнены тем, что счетчик будет собираться с использованием двух различных типов триггеров (дополнительно к JK-триггеру с инверсным динамическим входом будет применен D-триггер с прямым динамическим входом). 1). Особенности проектирования счетчиков на триггерах с прямым и инверсным динамическим входом изложены в конспекте (см. первую лекцию по счетчикам). 2). D-триггер необходимо преобразовать в T-триггер, т.е. перевести его в счетный режим (также см. конспект).

Выполнение работы:

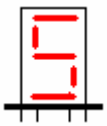
а) Получите задание: модуль счета M, а также набор триггеров (в задании последовательность триггеров в счетчике будет задана в виде, например "J-D-D-J"). Это означает, что крайние триггеры 0 и 3 - JK типа, а два средних 1 и 2 это D - триггеры,



не забудьте подключить, незадействованные входы ~S к логической "1", например таким образом:



б) заполните таблицу состояний и соответствующую ей таблицу Карно,
в) вычислите алгебраическое уравнение,
г) запустите Electronic WorkBench (см. п.5 лабораторной работы №20) и соберите свою схему аналогично рис.23-2. К счетному входу счетчика подключите младший разряд генератора слов (и переведите его в пошаговый режим суммирующего счетчика см. п. 5.3 лаб. работы №20),
д) для контроля состояний используйте 7-ми сегментный индикатор с встроенным преобразователем кода,



е) смену состояний в соответствии в заданием предъявите преподавателю.
ж) отключите обратную связь, удалив проводник Yoc из схемы, подключите освободившиеся входы ~R к логич. "1" и переделайте суммирующий

счетчик в вычитающий (2 способа приведены в конспекте), результат предъявите преподавателю.

СВОДНЫЙ ПЕРЕЧЕНЬ ВОПРОСОВ ДЛЯ ЗАЩИТЫ ЛАБОРАТОРНЫХ РАБОТ

Вопросы к работе №1:

1. Дать описание отдельных команд ассемблера, использующихся в программах.
2. Уметь комментировать каждую команду обеих программ.

Вопросы к работе №2:

0. Назначение, состав и принцип действия ППИ**.**
1. Объяснить работу схемы подключения ППИ к МП-системе (интерфейсная и функциональная части схемы).
2. Расчет адресов портов ввода/вывода.
3. Выполнение команд IN XX и OUT XX.
4. Способ вывода символов на дисплей. "Бегущий ноль". Для чего нужны электронные ключи и Rогр.?
5. Назначение отдельных битов управляющего байта ППИ.
6. Уметь комментировать каждую команду программы вывода на дисплей.

Вопросы к работе №3:

0. Назначение, состав и принцип действия ПИТ**.**
1. Объяснить работу схемы подключения ПИТ к МП-системе (интерфейсная и функциональная части схемы).
2. Расчет адресов портов ввода/вывода.
3. Назначение отдельных битов управляющего байта и режимы работы ПИТ.
4. Схема таймера для обратного отсчета времени с подачей звуковых отметок.
5. Расчет управляющих байтов и коэффициентов деления Ni каждого счетчика.
6. Уметь комментировать каждую команду программы запуска таймера.

Вопросы к работе №4:

1. Объяснить работу схемы подключения консоли к МП-системе.
2. Что такое программа-монитор.
3. Расчет адресов портов ввода/вывода и управляющего байта ППИ.
4. Объяснить работу монитора по блок-схеме.
5. Уметь комментировать каждую команду программы передачи

Вопросы к работе №5:

0. Назначение, состав и принцип действия PCI**.**
1. Объяснить работу схемы подключения USART к МП-системе.
2. Типовая последовательность битов в передаваемом кадре для асинхронного режима работы.
3. Расчет адресов портов ввода/вывода.
4. Назначение отдельных битов инструкции режима, команды управления и байта состояния УСАПП.

ПРИМЕРЫ ПРОГРАММИРОВАНИЯ ВНЕШНИХ УСТРОЙСТВ ДЛЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ
ЗАДАЧА 1. КОДОВЫЙ ЗАМОК

По приведенной схеме (рис. I-1)кодового замка и фрагменту программы укажите: а) HEX код содержимого ячейки памяти cb, б) адрес порта PB, в) что выполняет фрагмент программы, начиная с метки M1?: 0 – подает сигнал тревоги, 1 – блокирует замок, 2 – открывает замок, 3 – вызывает опергруппу. PPI работает в режиме 0.

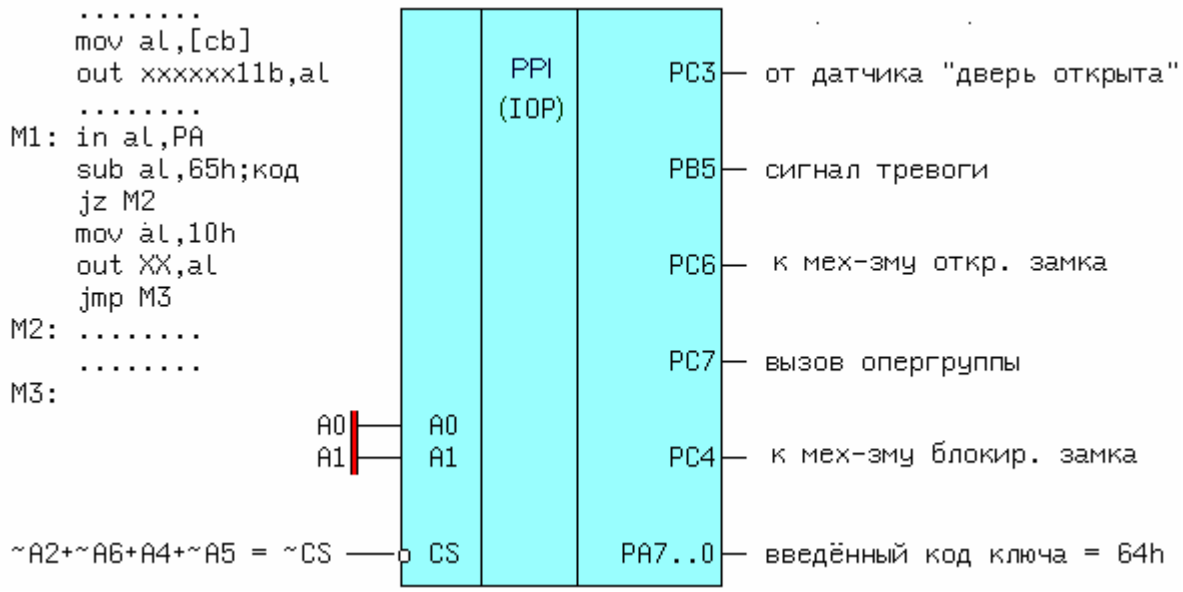


Рис. I-1 Кодовый замок

РЕШЕНИЕ

а) Для ответа на первый вопрос неплохо бы иметь карту памяти с символическим именем одной из ячеек - cb. Так как этого нет, то исследуем первые две команды фрагмента программы. Содержимое cb загружается в аккумулятор AL и затем выводится командой OUT в порт с двоичным адресом xxxxxx11 (b - указывает на двоичную запись, 6 бит нам пока не известны). На рисунке присутствует только одно ВУ, а именно PPI. Два младших бита шины адреса (ША) равные A1,A0 = 11 попадают на входы PPI A1,A0 - это свидетельствует, что содержимое AL будет "истолковано"

PPI, как управляющий байт. Не путайте входы PPI A1,A0 и биты ША A1,A0. В некоторых задачах входы A1,A0 микросхем могут соединяться с другими линиями ША. Для режима 0 управляющий байт PPI имеет следующий формат:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	PA	PCст	0	PB	PCмл

Значения битов PA, PB, PCст, PCмл определяются требуемым направлением передачи информации через один из указанных портов (половины порта PC могут настраиваться на ввод или вывод раздельно). Из рисунка следует, что порт PA предназначен для ввода кода ключа, следовательно, бит PA=1. Из младшей половины порта PC задействована только линия PC3 (по ней передается сигнал от датчика "дверь открыта"), поэтому бит PCмл = 1. Анализ линий портов PB,PCст показывает, что они задействованы на вывод, то есть биты PB и PCст равны 0. Поэтому ответом на первый вопрос будет код 91(HEX) = 10010001.

б) Адрес порта PB найдем из условия, что при выполнении команд IN или OUT обращенных к PPI, сигнал на входе ~CS должен быть активным, т.е. равным 0, т.к. вход ~CS инверсный. Сигнал на этом входе задан в виде уравнения неполного дешифратора, связывающего отдельные биты шины адреса (ША) по ИЛИ. Результат ИЛИ равен нулю, если ВСЕ входные сигналы ИЛИ также равны нулю. Часть битов ША использована в уравнении в инверсном виде, поэтому значения этих битов д.б. равны 1. Кроме того для порта PB биты A1,A0 = 01.

A7	A6	A5	A4	A3	A2	A1	A0
X	1	1	0	X	1	0	1

Незадействованные (don't care) биты заполняем любыми значениями, например нулями и находим один из вариантов ответа на второй вопрос (адрес порта PB = 01100101 = 65(HEX).

в) Какие действия выполняет PPI, можно узнать, анализируя остальной код фрагмента программы. Команда IN AL,PA что-то вводит в аккумулятор из порта с символическим именем PA. Нетрудно догадаться, что это имя относится к одноименному порту, и следовательно вводится набранный код ключа. В следующей команде введенный код вычитается или сравнивается (что одно и тоже) с каким-то числом. Не нужно быть Ш.Холмсом, чтобы догадаться, что это число - истинное значение кода ключа (в нашем случае оно совпадает с введенным).

В этот момент для варианта, когда коды совпадают - ответ ясен, однако рассмотрим решение подробно, т.к. в случае несовпадения кодов (в нашем случае 65 <>64) существует несколько альтернатив. Итак, обходим

команду JZ M2, как не соответствующую нашему случаю. Следующие две команды выводят код 10(HEX) = 00010000(BIN) в порт XX. Из восьми битов этого кода какое-то действие произведет, скорее всего, бит D4=1. Из приведенных на рисунке портов этот бит D4 может попасть только на линию PA4 (где ему делать нечего, т.к. порт PA настроен на ввод кода ключа) и на линию PC4 которая, как видно из рисунка передает сигнал на механизм блокировки замка. Третьим ответом будет: 1.

ЗАДАЧА 2. ДЕШИФРАТОР АДРЕСА

Назовите возможный адрес порта XX внешнего устройства ВУ4 и что в него будет выведено после выполнения приведенного фрагмента программы (рис. I-2)?

```
src EQU ab19h XOR 9770
..........
sub bh,bh
mov bl,0
mov ax,[src + bx]
xchg ah,al
out XX,al
..........
АДРЕС ¦ ДАННЫЕ
3c69  ¦ 20
3c6a  ¦ 5e
3c6b  ¦ 74
3c6c  ¦ f7
3c6d  ¦ 58
3c6e  ¦ 9e
3c6f  ¦ 7a
3c70  ¦ 4d
3c71  ¦ 08
3c72  ¦ 4f
3c73  ¦ cb
```

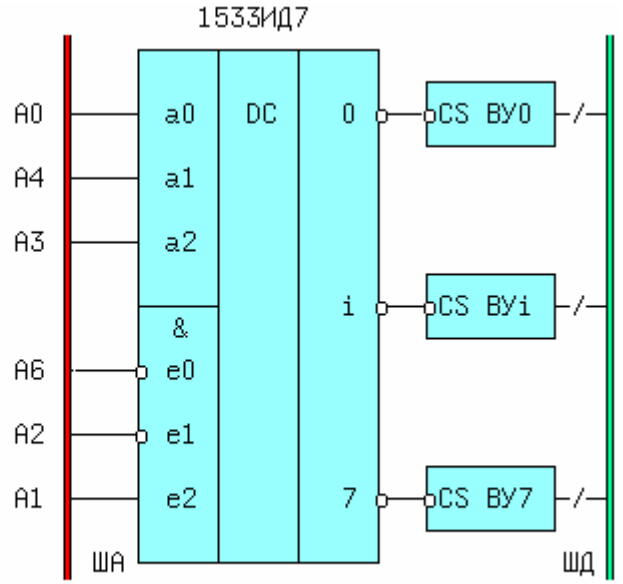


Рис. I-2 Дешифратор адреса

РЕШЕНИЕ

а) Ответ на первый вопрос нетрудно найти, зная, что адрес XX появится на шине адреса (ША) во время выполнения команды OUT. По условию задачи этот адрес должен относиться к ВУ4 и следовательно на 4-м выходе дешифратора должен появиться активный сигнал (в нашем случае - нулевой т.к. выходы дешифратора инверсные). В соответствии с определением дешифратора на информационных входах DC должен быть подан двоичный код активного выхода т.е. a2,a1,a0 = 100(BIN) = 4. При этом на разрешающие входы должны быть активизированы e2 * e1 * e0 = 1. Входы e1 и e0 - инверсные, поэтому сигналы на них д.б. равны 0 (~e1=~e0=0). Сигнал e2=1. Подставляя найденные значения в соответствующие биты ША и заполняя незадействованные биты чем угодно, например нулями, найдем адрес = 0A(HEX).

A7	A6	A5	A4	A3	A2	A1	A0
X	0	X	0	1	0	1	0

б) Для ответа на второй вопрос рассмотрим фрагмент программы. Команда SUB BH,BH обнуляет старшую половину регистра BX, а следующая команда заносит в его младшую половину 0. Следовательно, после выполнения первых двух команд содержимое BX равно 0. Инструкция MOV AX,[SRC+BX] пересылает два байта в регистр AX из двух последовательно расположенных ячеек памяти с адресами SRC+BX и SRC+BX+1. Содержимое BX уже известно, осталось найти значение константы с именем SRC. Директива EQU присваивает SRC значение двух числовых констант, объединенных операцией XOR (исключающее ИЛИ).

Результат XOR (как и любой другой двухоперандной логической команды) легче всего найти, записав константы в двоичном виде друг под другом и произведя побитовые операции по правилу - результат XOR равен 1, если равен 1 только один аргумент.

ab19 = 1010 1011 0001 1001
9770 = 1001 0111 0111 0000

_____
0011 1100 0110 1001 = 3c69 результат

Отсюда следует, что SRC+BX = 3c69+0=3c69 и по команде MOV AX,[SRC+BX] в двухбайтовый регистр AX пересылаются подряд два байта из двух ячеек памяти с адресами 3c69 и 3c6a, то есть байты 20 и 5e. Причем действует правило - байт по старшему адресу (5e) помещается в старшую половину регистра AX (AH), а байт по младшему адресу (20) - в регистр AL. Инструкция XCHG AH,AL меняет местами содержимое AH и AL, а команда OUT XX,AL выводит в ВУ4 байт 5e.

ЗАДАЧА 3. ТАЙМЕР

По приведенной схеме (рис. I-3) программируемого таймера 580ВИ53 и фрагменту программы укажите: 1. Возможное значение адреса порта XX активного счетчика в HEX – коде. 2. Номер этого счетчика. 3. Значение частоты в Гц на его выходе для режимов 2 и 3 или длительность импульса – в милисекундах для режима 1. 4. Номер временной диаграммы, условно соответствующей задаче. Начальные значения: (AL)=81h, (BL)=04h, (CL)=FEh. На все входы CLK подается сигнал с частотой Fclk=1000Гц.

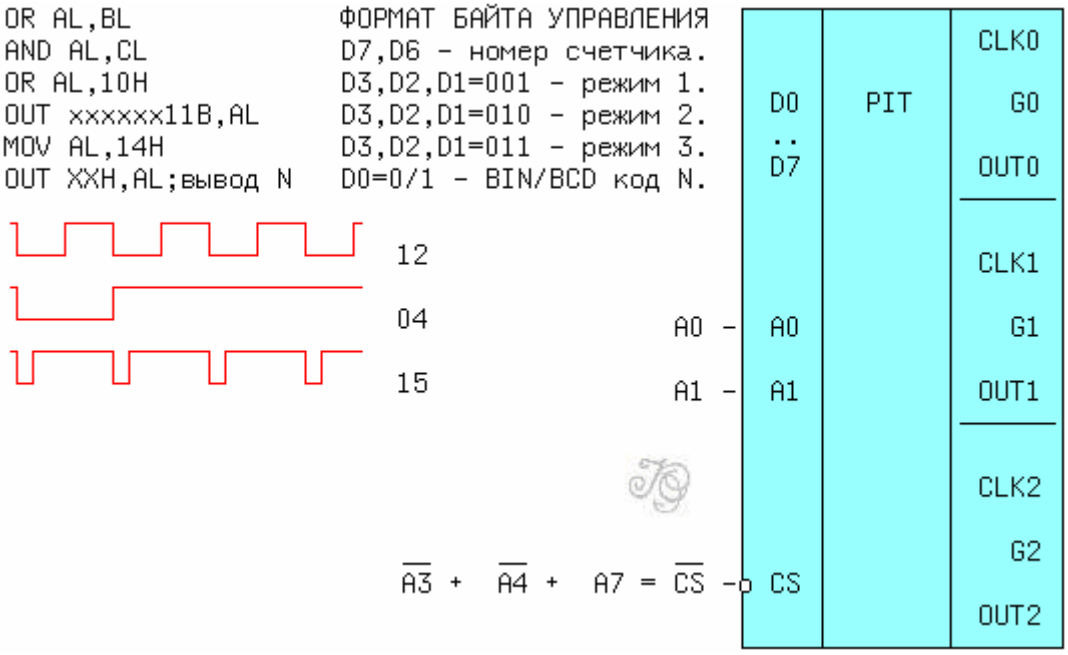


Рис. I-3 Программируемый интервальный таймер

РЕШЕНИЕ

а) Ответы на 2,3 и 4 вопросы можно найти, определив управляющий байт PIT. Команда OUT xxxxxx11b,AL выводит управляющий байт в PIT, о чем свидетельствуют два младших бита адреса A1,A0=11. Определим содержимое аккумулятора на момент выполнения этой команды, для чего необходимо выполнить три логические команды (OR)ИЛИ, (AND)И и снова ИЛИ.

81H = 1000 0001 (al)
04H = 0000 0100 (bl)
_____ OR
85H = 1000 0101 (al) новое значение

85H = 1000 0101 (al)
FEH = 1111 1110 (cl)
_____ AND
84H = 1000 0100 (al) новое значение

84H = 1000 0100 (al)
10H = 0001 0000
_____ OR
95H = 1001 0100 (al) новое значение (управляющий байт)

Отсюда находим ответ на второй вопрос: номер счетчика D7,D6=10(Bin)=2.

Так как биты D3,D2,D1=010, то счетчик работает во втором режиме (генератор периодической последовательности со скважностью Q>2) и

ответом на 4-ый вопрос будет 15 (номер временной диаграммы для этого режима).

Для второго режима выходная и входная частоты связаны соотношением Fout = FclK/N. Коэффициент деления N в программе записывается в счетчик (во второй) в последних двух командах, причем записывается одним младшим байтом, так как биты D5,D4=01. 14 можно истолковать как 14 в двоично-десятичном коде или как $1*16^1 + 4*16^0 = 20$ в 16-ном. PIT "истолкует" это число как двоичное, так как бит D0 управляющего байта = 0. Следовательно коэффициент деления N=20(DEC) и частота на выходе второго счетчика Fout2 = 1000Гц/20 = 50Гц. Это ответ на второй вопрос.

Как и в задаче 1 адрес порта можно вычислить, анализируя уравнение на входе ~CS. Биты адреса A4=A3=1, бит A7=0, для второго счетчика биты A1,A0=10(BIN)=2, остальные биты заполняем чем угодно (например нулями) и получаем один из возможных адресов второго счетчика: 1A(HEX).

ЗАДАЧА 4. УПРАВЛЕНИЕ ЦАП

1. Вычислите ряд значений на выходе устройства #/^ при выполнении блока команд от метки OUTP до метки EXIT (рис. I-4). Полярность Uвых – положительная. 2. Определите порт вывода PX (PA,PB или PC).

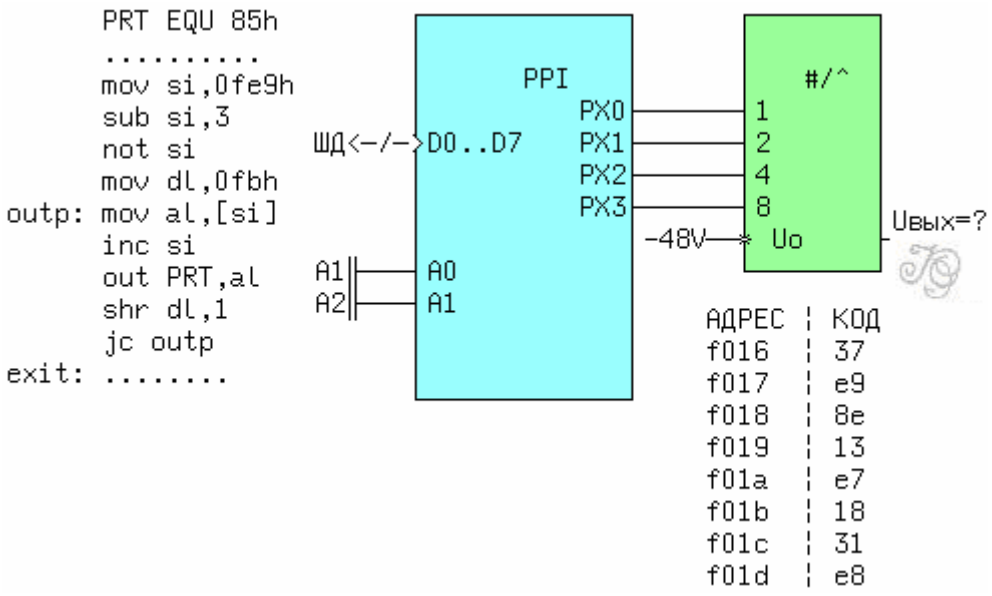


Рис. I-4 Управление ЦАП

РЕШЕНИЕ

а) Определяем, что устройство #/^ - есть цифро-аналоговый преобразователь, у которого Uвых = $(-Uo*D)/2^n$, где n - число разрядов ЦАП (n=4,Uo=-48v), D- десятичный эквивалент двоичного кода на входах ЦАП. Поэтому Uвых = (3*D) вольт. Первые три команды дадут (SI)=0FE9-3=0FE6

и после инверсии (SI)=F019. Четвертая команда засылает в регистр DL значение FB(HEX)=1111 1011(BIN).
Далее отмечаем, что циклическое выполнение команд

```
outp: mov al,[si]
      .....
      shr dl,1
      jc outp
```

выполняется пока команда SHR DL,1 "выталкивает" во флаг переноса CF единицу и закончится, когда очередной сдвиг вправо содержимого DL "вытолкнет" во флаг переноса 0 и команда JC "перейти если есть перенос" не будет выполнена. Так как в коде 11111011-->CF ноль попадет в CF только при третьем проходе, то и весь цикл будет выполняться 3 раза.
Команды

```
outp: mov al,[si]
      inc si
      out PRT,al
```

выведут в порт с адресом PRT содержимое трех последовательных ячеек памяти, начиная с адреса F019, то есть 13,e7,18 (HEX). Как видно из рисунка, на входы четырехразрядного ЦАП попадут только четыре младших бита (3,7,8) через линии PX0..PX3.

Отсюда следует, что в цикле команд, начиная с метки outp, инструкцией OUT PRT,AL на выходе ЦАП будут сформированы три значения напряжения: 3*3=9 вольт при первом проходе и 3*7=21 и 3*=24 вольт при остальных проходах.

б) Адрес порта вывода известен из директивы PRT EQU 85H (1000 0101) в котором биты ША A2,A1=10(BIN). Эти биты попадают на входы A1,A0 PPI (не путать линии ША A2,A1 и входы A1,A0). Когда код на входах A1,A0=10, то активизируется порт PC и вывод идет через него.

ЗАДАЧА 5. СИСТЕМА ОБРАБОТКИ ДАННЫХ

К четырем портам PPI подключены три ВУ (в том числе некий прибор). По приведенной схеме (рис. I-5) и фрагменту программы укажите HEX значение байта управления и адрес ячейки памяти, где он хранится. Формат управляющего байта: 1 0 0 PA PCст 0 PB PCмл. Исходные данные: (AL)=04h, (BH)=9ch, (DH)=2ah.

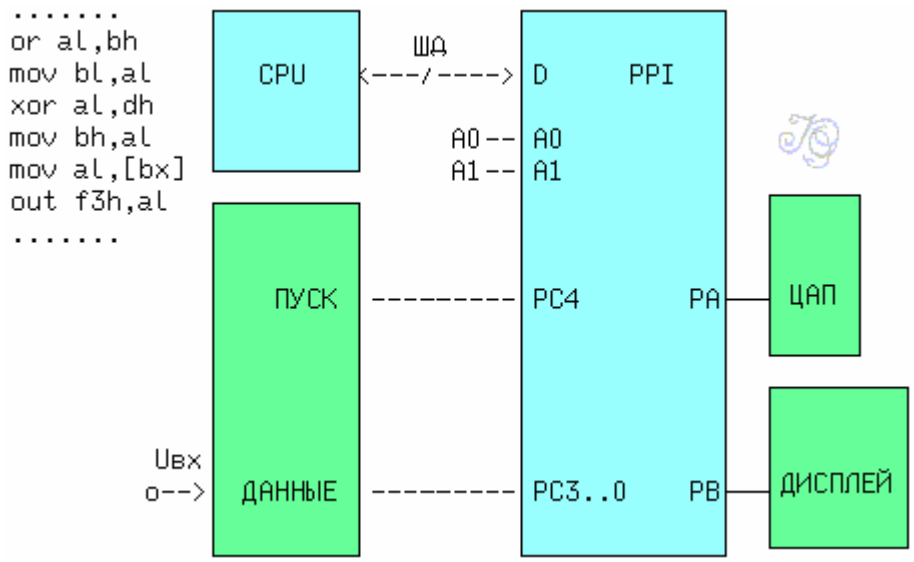
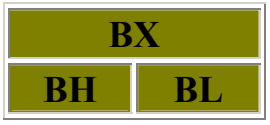


Рис. I-5 Система обработки данных

РЕШЕНИЕ

а) Найдем сначала байт управления. Цифроаналоговый преобразователь принимает код через порт PA, который поэтому должен быть настроен на вывод. Следовательно, бит PA = 0. Дисплей отображает код, выводимый через порт PB, который также настраивается на вывод. Поэтому бит PB = 0. Слева к ППИ подключен некий прибор, который повидимому измеряет напряжение (о чем свидетельствует "Uвх o-->") и следовательно порт "ДАННЫЕ" предназначен для считывания кода Uвх. Тогда младшая половина порта PC3..0 должна быть настроена на ввод и бит PCмл = 1. Относительно вывода "ПУСК" можно допустить, что прибор либо что-то через ППИ запускает в микропроцессоре, либо микропроцессор сам запускает прибор на измерение. Вторая версия явно предпочтительнее, так как опирается на конкретную схему, а не на догадки. Поэтому старшая половина порта PC должна быть настроена на вывод (бит PCст = 0) и управляющий байт = 1000 0001 = 81(HEX).

б) Анализ операнда F3(1111 0011) команды OUT показывает, что 2 младших бита адреса A1,A0 = 11, поэтому содержимое аккумулятора является для ППИ управляющим байтом. Инструкция MOV AL,[BX] извлекает его из ячейки памяти с адресом находящимся в регистре BX. Остается найти содержимое этого регистра, принимая во внимание, что BX состоит из двух половин BH,BL.



Содержимое BH на момент выполнения MOV AL,[BX] равно B6, содержимое BL = 9C (нахождение (BH) и (BL) производится аналогичным

третьей задаче способом, с учетом того, что вместо AND здесь операция XOR и порядок команд другой). Следовательно, адрес =B69C.

ЗАДАЧА 6. ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС

Что осуществляется? Передача или прием байта данных? Найдите HEX код переданного (принятого) символа по приведенной диаграмме (рис. I-6), а также возможный адрес порта. Начальные значения: (AL)=0bh, (DH)=16h, (BH)=17h.

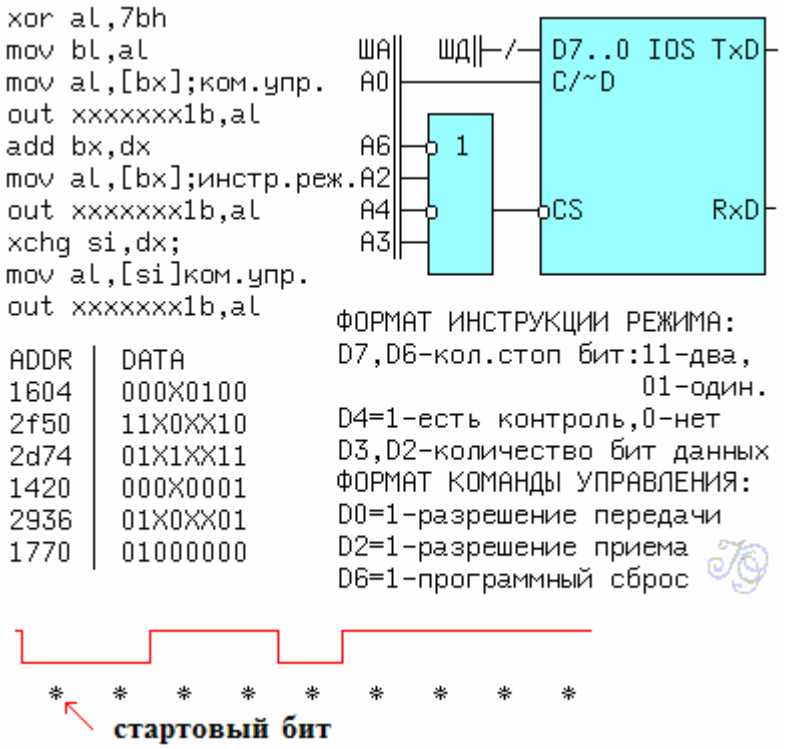


Рис. I-6 Последовательный интерфейс

РЕШЕНИЕ

На рисунке приведена схема подключения УСАПП. Находим, что [BX] = 1770 на момент третьей команды управления, поэтому из ячейки памяти с этим адресом будет извлечен код 0100 0000 (бит D6=1), что свидетельствует о программном сбросе УСАПП. Команда ADD BX,DX складывает 1770(HEX) + 1604(HEX), на рисунке приведены начальные значения половин регистра DX (16 и 04). Результат сложения (2D74) является адресом инструкции режима (01X1XX11), о чем свидетельствует комментарий ко второй команде MOV AL,[BX].

D7	D6	D5	D4	D3	D2	D1	D0
0	1	X	1	X	X	1	1

а) Найдем код символа. Число информационных битов указывается в битах D3,D2, но они как назло в инструкции не указаны (D3,D2=XX). Поэтому их число найдем косвенным способом. Бит D4=1, поэтому независимо от бита D5, известно, что контроль присутствует. Биты D7,D6=01 свидетельствуют, что количество стоп-битов = 1. Мысленно отбрасывая служебные биты (один крайний левый - стартовый и два крайних правых - бит контроля и стоп-бит) оставляем 6 средних информационных бит (011011). Так как на временной диаграмме они следуют "ногами вперед" слева младший, а справа старший разряд, развернем их в нормальной для текстовой записи последовательности (слева старший разряд, справа - младший). Полученный код 11 0110(BIN) = 36(HEX), является ответом на второй вопрос.

б) Далее, команда XCHG SI,DX обменивает содержимое регистров SI и DX. Начальное значение SI в задаче не приводится, но как следует из последних двух команд оно и не нужно. В момент выполнения предпоследней команды MOV AL,[SI] новое значение [SI] равно старому значению [DX], которое было равно 1604(HEX). Из ячейки памяти с этим адресом будет извлечена команда управления (000X0100). Бит D2=1 свидетельствует, что УСАПП работает в качестве приемника. Это является ответом на первый вопрос.

в) Для нахождения возможного адреса порта (данных или байта управления в задаче не уточняется), как обычно необходимо проанализировать схему дешифратора (в данной схеме неполного и выполненного на элементе ИЛИ).

ЗАДАЧА 7. КЛАВИАТУРА

По приведенному фрагменту программы (рис. I-7) вычислите HEX код нажатой клавиши (содержимое регистра DL). Содержимое регистра BL равно коду на выводах порта PB, а (CL) = коду на выводах порта PC. Начальное значение (DL)=0.

РЕШЕНИЕ

Из рисунка видно, что нажата клавиша в верхнем левом углу матрицы 4*4. В этом случае схема зафиксировала 0 в верхней строке и левом столбце матрицы. Эти нули соответственно располагаются на линиях портов PB1 и PC2. В приведенном фрагменте программы, оформленном в виде подпрограммы содержатся два "бесконечных" цикла, из которых предусмотрены выходы "по условиям".

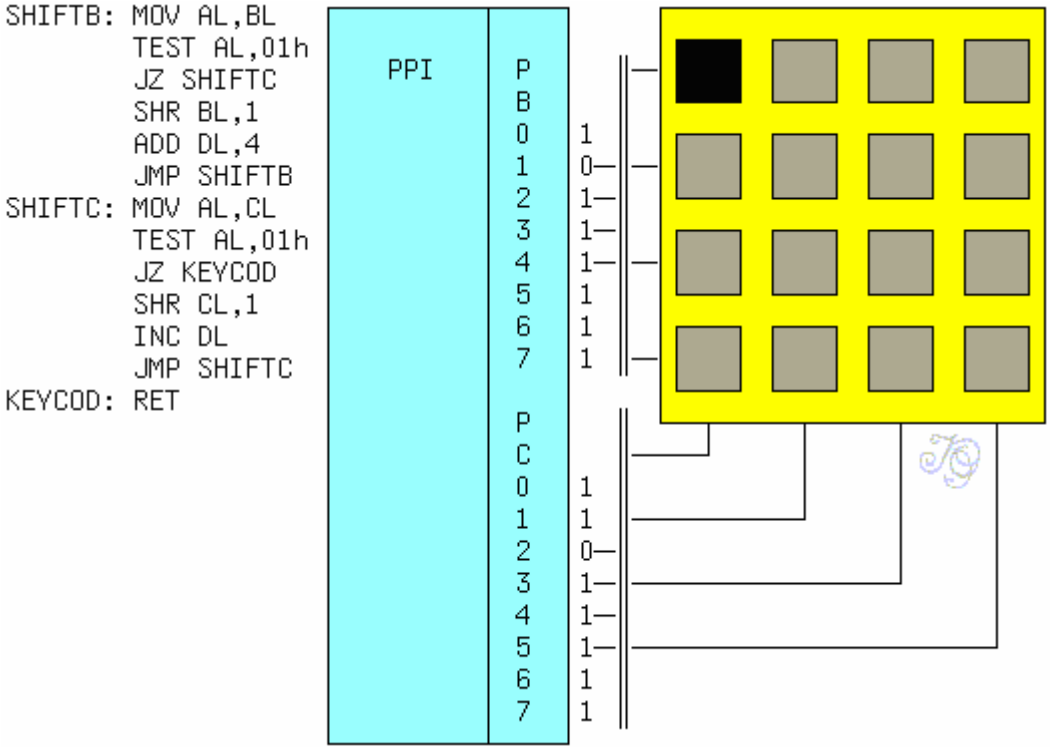


Рис. I-7 Клавиатура

В первом цикле от метки SHIFTB: ... до комады JMP SHIFTB содержимое регистра BL проверяется командой TEST (логическое поразрядное И) с помощью маски 01(HEX)=0000 0001(BIN). При ненулевом результате выполняются остальные команды этого цикла, при нулевом (когда маска и содержимое регистра BL совпадают) инструкция JZ SHIFTC осуществляет выход из указанного цикла. Отсюда видно, что цикл будет выполняться до тех пока 0 в регистре BL не окажется в позиции занятой единицей маски (то есть один раз).

00000001 маска
11111101 (BL) до сдвига вправо
-------- test
00000001 - результат не нулевой
00000001
01111110 после сдвига вправо shr
-------- test
00000000 - результат нулевой

Таким образом, при однократном проходе первого цикла (однократное выполнение команды ADD DL,4) в регистре DL будет храниться 4.

Второй цикл SHIFTC: ... JMP SHIFTC аналогичен первому, но выполняться будет два раза.

00000001 маска
11111011 (CL) до сдвига вправо
-------- test
00000001 - результат не нулевой
00000001
00111110 после двух сдвигов вправо shr
-------- test
00000000 - результат нулевой

Таким образом, при двукратном проходе второго цикла (двукратное выполнение команды INC DL) в регистре DL будет храниться значение 4 + 2 = 6, которое и будет ответом на задачу.

ЗАДАЧА 8. УПРАВЛЕНИЕ ПРИЗМОЙ СПЕКТРОМЕТРА

Чему равны HEX значения байтов xx.., YY и P1? На сколько градусов повернется призма спектрального прибора после выполнения приведенного фрагмента программы (рис. I-8)? Призма повернулась за время T1=1/4 сек.

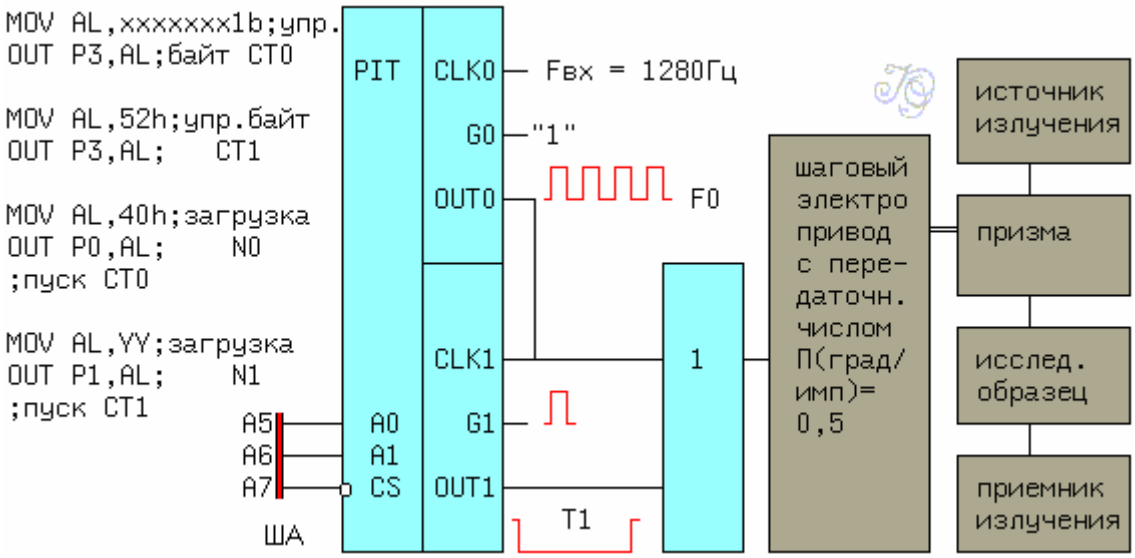


Рис. I-8 Управление примой спектрометра

РЕШЕНИЕ

а) Вычислим значение байта xx..x1. Из комментария следует, что это управляющий байт счетчика CT0. Для каждого счетчика два старших бита определяют его номер. Для CT0 биты D7,D6 = 00. Временная диаграмма на выходе OUT0 счетчика CT0 соответствует режиму 3 (генератор периодической последовательности со скважностью равной двум, то есть длительность импульса и паузы равны). Поэтому биты D3,D2,D1 для третьего режима = x11 (011 или 111). Бит D0=1 по условию (xxxxxxx1), то есть коэффициент деления N0 записывается в счетчик двоично-десятичным кодом. Загрузка коэффициента деления N0 производится, как видно из

программы, одним байтом, но каким? Если загрузка N0 производится старшим байтом, то коэффициент деления = 4000(DEC), если младшим, то = 0040(DEC). Ответ на этот вопрос получим проанализировав общий коэффициент деления N двух последовательно включенных счетчиков CT0 и CT2. N = N0*N1 = T1/(T0) = T1/(1/F0) = 0.25/(1/1280) = 320(DEC), что значительно меньше, чем 4000. И следовательно запись N0 в CT0 должна производиться только одним младшим байтом равным 40(DEC). Поэтому биты D5,D4 управляющего байта равны 01(BIN).

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	X	1	1	1

Окончательно, находим управляющий байт: 0001 0111 = 17(HEX) или 0001 1111 = 1F(HEX).

б) YY, как следует из комментария, является коэффициентом деления N1, так как N = 320(DEC) и N0 = 40(DEC), то из N = N0*N1 находим ответ YY = N1 = 8.

в) Значение P1 в команде OUT P1,AL является адресом первого счетчика и появляется на ША в момент выполнения этой команды. Причем, естественно, на входе ~CS должен быть 0 (сигнал на линии ША A7=0), а на входах PIT A1,A0 должна быть комбинация соответствующая номеру этого счетчика, то есть 01(BIN). Следовательно биты ША A6,A5 также должны быть равны 01. Остальные незадействованные биты ША, как обычно заменяем чем угодно, например нулями.

A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	X	X	X	X	X

Тогда одним из возможных ответов будет P1 = 80(HEX)

г) Осталось найти, на сколько градусов повернется призма. Для этого необходимо вычислить число импульсов прошедших через схему И, выполненную на логическом элементе ИЛИ.

Длительность T1=0.25 сек по условию. Длительность T0 на выходе OUT0 найдем зная F0 = Fвх/N0 = 1280Гц/40 = 32Гц. То есть T0 = 1/F0 = 1/32 сек. Число импульсов на выходе ИЛИ равно T1/T0 = 0.25/(1/32) = 8 импульсов. Призма поворачивается на 0.5 градуса за один импульс, следовательно, за 8 импульсов она повернется на 4 градуса.

ЗАДАЧА 9. СИСТЕМА ОХРАНЫ

При пересечении невидимого луча оптикоэлектронного сторожевого устройства (рис. I-9) должен раздаться сигнал тревоги. Будет ли услышан сигнал? Два исправления дать в том порядке, в каком они следуют. В третьем ответе назовите номер счетчика.

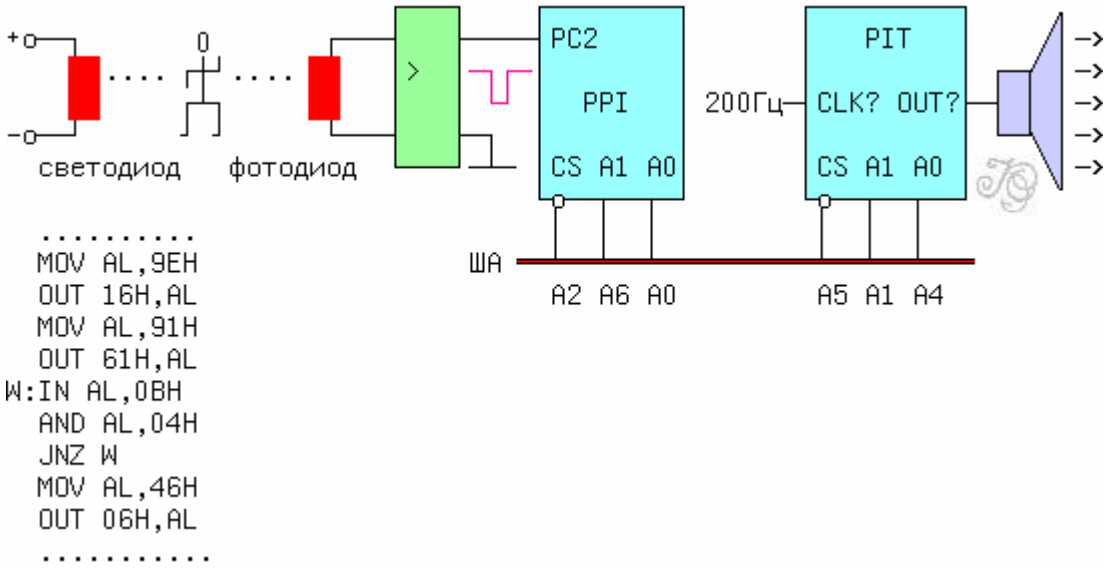


Рис. I-9 Система охраны

РЕШЕНИЕ

Рассмотрим первые две команды. Адрес 16(HEX) = 0001 0110(BIN) появится на ША в момент выполнения второй команды.

A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	1	0	1	1	0

Так как бит A2=1, а A5=0, то будет активизирован вход ~CS программируемого интервального таймера PIT. Причем биты A4,A1 ША поступая соответственно на входы PIT A0,A1 свидетельствуют, что по команде OUT 16H,AL в ПИТ записывается управляющий байт 9E(HEX) = 1001 1110(BIN).

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	1	1	1	0

Расшифровка управляющего байта дает следующую информацию: D7,D6 = 10(BIN) = 2, значит для управления динамиком задействован второй счетчик (ответ на третий вопрос), запись N2 производится одним младшим байтом (D5,D4 = 01), используется третий режим (генерация

периодической последовательности - D3,D2,D1 = 111) и N2 представлен двоичным кодом (D0 = 0).
Вторые две команды выводят по адресу 61H = 0110 0001(BIN)

A7	A6	A5	A4	A3	A2	A1	A0
0	1	1	0	0	0	0	1

байт 91H = 1001 0001(BIN).

D7	D6	D5	D4	D3	D2	D1	D0
режим 0			PA	PCст	режим 0	PB	PCмл
1	0	0	1	0	0	0	1

Бит A2 шины адреса равен нулю, а бит A6 = 1, поэтому активизирован будет программируемый периферийный интерфейс - PPI (~CS=0). Бит PCмл управляющего байта показывает, что как и требуется в задаче линия PC2 настроена на ввод сигнала от фотоприемника.
Далее рассмотрим работу фрагмента программы:

```
W: IN AL,0BH;
   AND AL,04;
   JNZ W;
```

По-видимому этот цикл должен опрашивать состояние линии PC2 (если PC2 = 1, то луч не прерван, PC2=0 - свидетельствует о пересечении луча). Другой подходящей версии о назачении этого фрагмента придумать трудно. Тогда команда IN AL,0BH должна считывать код из порта PC (линия PC2). Рассмотрим адрес 0B(HEX) = 0000 1011(BIN).

A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	1	0	1	1

Бит A2=0, что активизирует ППИ для обращения к нему микропроцессора, но биты ША A6,A0 = 01, что соответствует чтению из порта PB, а не PC. Кроме того, бит A5 также равен 0, поэтому одновременно будет активизирован и PIT. Следовательно, адрес 0B является ошибочным. Для чтения из порта PC биты ША A6,A0 должны быть равны 10(BIN). Поэтому возможным адресом порта PC вместо 0B явится адрес с битами A6=1,A5=1,A0=0,A2=0, например 60(HEX). Это первое исправление и первый ответ.

В команде AND AL,04H с помощью маски 0000 0100 проверяется бит D2, который соответствует биту PC2, то есть ошибки здесь нет. Команда JNZ W(ait) производит переход на метку W "по не нулю", то есть этот цикл повторяется до тех пор пока на линии PC2 не появится ноль, соответствующий пересечению луча. Здесь тоже нет ошибки.
Осталось рассмотреть последние две команды:

MOV AL,46H
OUT 06,AL

Переход к этим командам произойдет, только если луч прервется и возникнет необходимость в подаче сигнала тревоги. Адрес 06(HEX) в команде OUT

A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	1	0

активизирует PIT, так как A5 = ~CSPIT = 0. Бит A2 = ~CS PPI = 1, то есть PPI пассивен. Биты ША A1,A4 равны битам PPI A1,A0 и равны 10(BIN) = 2 (это есть ответ на третий вопрос). Следовательно код 46(HEX) = 4*16 + 6 = 70(DEC) записывается в счетчик CT2. Так как последние две команды выполняются, как уже говорилось, только при необходимости подачи сигнала, то ничего другого как запуск второго счетчика на генерацию сигнала тревоги не придумать. Как известно в третьем режиме генерация начинается после записи коэффициента деления N2 в счетчик (при очевидном значении сигнала GATE = 1, на рисунке поэтому даже не показанным). Следовательно N2 = 46(HEX) = 70(DEC) и есть коэффициент деления. Тогда частота сигнала на выходе динамика будет равна 200Гц/70 = 2.85Гц и услышана никем не будет. Это и есть вторая ошибка. Для ее исправления во втором ответе нобходимо привети такое значение N2, чтобы частота на выходе CT2 находилась в пределах 20Гц <= Fout2 <= 16000Гц. Например: N2 = 5 (что и является одним из возможных ответов по исправлению второй ошибки).

ЗАДАЧА 10. СИСТЕМА ИНДИКАЦИИ

Найдите значения XX, YY, ZZ, VV и WW в программе (рис. I-10), чтобы после ее выполнения на линейке i=3 из 8-ми светодиодов можно было прочитать код ASCII код ‘Q’. Будьте внимательны! Посторонний человек должен прочитать код именно этого символа, а не что-то другое. Например, вместо кода 3A код 5C.

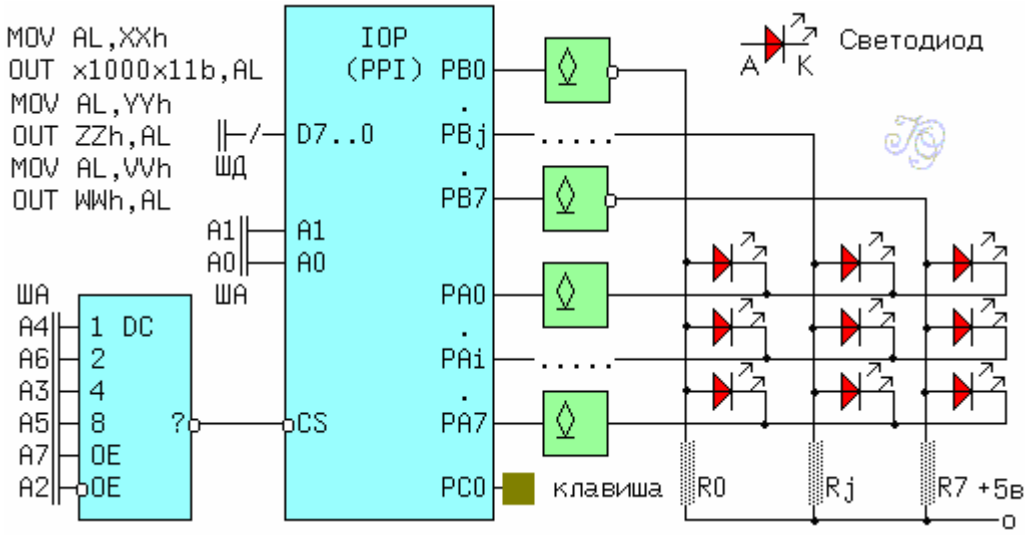


Рис. I-10 Система индикации

Справка: 1) ASCII коды символов ‘A’ … ‘Z’ находятся в пределах 41h … 5ah. 2) Назначение клавиши нам с вами неизвестно, но учитывать ее необходимо.

РЕШЕНИЕ

Задача не для слабонервных - нужно привести 6 ответов.

а) Найдем сначала XX. В команде OUT x1000x11B,AL два младших бита адреса A1,A0=11, поэтому XX является управляющим байтом. Из рисунка и справки следует, что порты PA и PB должны быть настроены на вывод,

D7	D6	D5	D4	D3	D2	D1	D0
режим 0			PA	PCст	режим 0	PB	PCмл
1	0	0	0	X	0	0	1

а младшая половина порта PC - на ввод. Одно из двух возможных решений (бит X=0) будет XX = 81(HEX).

Далее рассмотрим схему подключения светодиодов. Для свечения диода в строке (линейке) с номером "i" и в столбце с номером "j" необходимо на анод подать высокий потенциал, а на катод - низкий. Выходы порта PB подключены к анодам через инверторы, а выходы порта PA - к катодам через повторители, поэтомы на выходе PBj должен быть 0, а на выходе PAi также 0.

Следующие 4 команды выводят через порт PB инверсный код символа и через порт PA - логический ноль на третью линейку (на остальные линейки - единицы). Порядок вывода (при оценке ответа) не играет роли, поэтому примем, что команды с операндами УУ,ZZ выводят код символа в порт PB, а команды с операндами VV,WW в порт PA.

б) Найдем код УУ, выводимый через порт PB. ASCII код буквы Q равен 51(HEX) = 0101 0001(BIN). Как уже говорилось, перед выводом его необходимо проинвертировать - 1010 1110. Однако этого мало, так как из схемы видно, что старший бит кода PB7 выводится на крайний правый светодиод линейки, а младший PB0 - на крайний левый. Обычно (за некоторыми региональными исключениями) считываем мы код с линейки (так же, как читаем) слева - направо. Поэтому в приведенной схеме в коде при выводе в линейку необходимо еще и переставить биты (зеркально отобразить). Тогда окончательно УУ = 0111 0101 = 75(HEX).

в) Раз мы начали вывод с порта PC, то ZZ является его адресом. Для PB на входах A1,A0 должна быть комбинация 01, и следовательно, в момент выполнения команды OUT ZZH,AL код на линиях ША A1,A0 также равен 01. ZZ = xxxxxx01. Где взять старшие 6 битов адреса? Из второй команды фрагмента программы - OUT x1000x01,AL (два младших бита уже соответствуют порту PB). Но два бита A7,A2 по-прежнему не определены. Обращаем внимание на схему дешифратора, в которой линия A7 подключена к прямому входу разрешения OE и следовательно A7=1, а линия A2 подключена к инверсному входу разрешения OE и следовательно A2=0.

A7	A6	A5	A4	A3	A2	A1	A0
1	1	0	0	0	0	0	1

Тогда ответом будет ZZ = C1(HEX).

г) Код VV мы решили выводить через PA, поэтому на 3-ю линию этого порта необходимо вывести 0, а на остальные - 1.

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	1	1	1

Поэтому VV = F7(HEX).

д) Адрес WW - это адрес порта PA и он отличается от адреса PB, который уже найден только двумя младшими битами A1,A0 = 00. WW = C0(HEX).

е) Осталось найти номер выхода дешифратора. Для этого определим значения на линиях ША подключенных к 4-м информационным (адресным) входам дешифратора и просуммируем весовые коэффициенты соответствующих входов.

A5	A3	A6	A4
8	4	2	1
0	0	1	0

По определению работы дешифратора номером активизированного выхода будет десятичный эквивалент двоичного кода на его адресных входах, т.е. двойка.

ЗАДАЧА 11. КОНТРОЛЛЕР ПРЕРЫВАНИЙ

В ответ на запрос, поступивший на вход IRi от ВУi устройство PIC (Рис. I-11) сформировало 3 байта CDXXCD, причем 8 8 сформированных схемой битов младшего адреса байта адреса равны 10x0x00. Определить: 1. Адрес подпрограммы обработчика прерывания ВУ, подавшего запрос. 2. Команду ICW1. PIC работает с МП 8085.

АДРЕС	КОД	АДРЕС	КОД
CDXX	C3(КОП JMP) ВУ3	3DC3	п/п
.....	3D ВУ3	.....	обраб.
.....	C3 ВУ3	.....	прерыв.
.....	.....	.....	.....
CDXX	C3(КОП JMP) ВУ6	333C	п/п
.....	33 ВУ6	.....	обраб.
.....	3C ВУ6	.....	прерыв.
.....	.....	.....	.....
CDXX	C3(КОП JMP) ВУ7	3C33	п/п
.....	3C ВУ7	.....	обраб.
.....	33 ВУ7	.....	прерыв.
.....	.....	.....	.....
CDXX	C3(КОП JMP) ВУ4	C33D	п/п
.....	C3 ВУ4	.....	обраб.
.....	3D ВУ4	.....	прерыв.
.....	.....	.....	.....

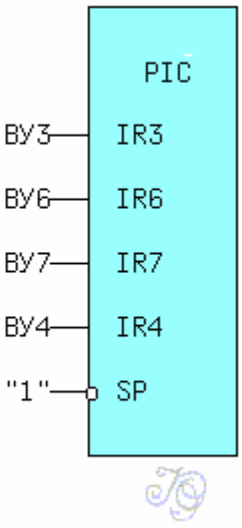


Рис. I-11 Контроллер прерываний

РЕШЕНИЕ

На рисунке приведен программируемый контроллер прерываний (PIC) к которому подключены 4 источника прерываний. Как известно в ассемблере МП 580-й серии код операции CD(HEX) соответствует команде CALL. Тогда CDXXCD расшифровывается , как CALL CDXX (вызов подпрограммы с адресом CDXX - не забываем, что байты адреса в коде и в исходном тексте переставляются). Для того, чтобы " окончательно подзапутать" решающего эту задачу старший байт адреса A15..A8 = CD(HEX) совпадает с кодом операции CD, а адреса обработчиков прерываний также имеют совпадающие байты. Ниже приведена таблица формирования контроллером младшего байта адреса XX(HEX) A7..A0 вызываемой подпрограммы. Для адресного интервала в 8 байтов между последовательно располагающимися адресами подпрограмм CALL УУXX, три старших бита команды инициализации контроллера прерываний ICW1 частично совпадают с номером ВУ подавшего запрос на прерывание.

вход запроса	Адресный интервал 4 байта								Адресный интервал 8 байт							
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
	3 старших бита ICW1			номер ВУ					3 старших бита ICW1							
IR7	A7	A6	A5	1	1	1	0	0	A7	A6	1	1	1	0	0	0
IRi	1	0	x	0	x	0	0	0	1	0	x	0	x	0	0	0
....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....	....
IR0	A7	A6	A5	0	0	0	0	0	A7	A6	0	0	0	0	0	0
											номер ВУ					

а) Для определения адреса подпрограммы источника прерывания (3DC3, 333C, 3C33 или C33D - приведенны на рисунке) необходимо найти номер ВУ (3,7,6,4) подавшего запрос и из соответствующей этому ВУ команды JMP ZZWW извлечь адрес перехода ZZWW, который и будет ответом на первый вопрос. Адресный интервал пока не известен, поэтому попробуем расшифровать номер ВУi (i=0x0) для 4-х байтового интервала и (i=x0x) для 8-ми байтового. Комбинация битов 0x0 дает два номера 000=0 и 010=2, но ВУ с такими номерами в схеме отсутствуют. Комбинация битов x0x для 8-ми байтового интервала дает четыре номера 000=0, 001=1, 100=4 и 101=5. Только ВУ4 присутствует в схеме. Команда C3C33D перехода к обработчику прерывания (C3 - код команды JMP) или JMP 3DC3 (байты адреса переставлены) дает ответ на первый вопрос: адрес обработчика = 3DC3.

б) Формат команды ICW1 приведен ниже: биты A7,A6,A5 - три старших бита младшего байта адреса обработчика (уже найдены), F - определяет, какой интервал ипользуется 8-ми байтовый (F=0) или 4-х байтовый (F=1).

D7	D6	D5	D4	D3	D2	D1	D0
A7	A6	A5	1	0	F	S	0

Из решения на первый вопрос уже известно, что номер ВУ 4 для заданных начальных условий (x0x=4), может располагаться только во

второй половине таблицы с адресным интервалом 8 байтов. Поэтому бит F=0. Бит S=1 свидетельствует, что используется один PIC, а S=0 - показывает, что контроллеров несколько. Из рисунка видно, что сигнал ~SP=1, следовательно, PIC в схеме один и бит S=1. Тогда код ICW1 = 101 10 0 1 0 = B2(HEX).

НЕКОТОРЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Шина - группа проводников (линий связи), имеющих похожее функциональное назначение или выполняющих общую функцию.
Строб - короткий импульс, выделяющий установившееся, свободное от помех значение информационного сигнала.
Системная шина - совокупность трех шин: шины данных (ШД), шины адреса (ША) и шины управления (ШУ).
КОП - однобайтовый код операции (первый байт любой команды)

ALU Арифметико-логическое устройство (АЛУ)
CE Chip Enable или Crystall Enable (тоже CS)
CMOS CoMpleMentary-syMMetry/Metal-oxide seMiconductor, комплементарная логика на транзисторах металл-оксид-полупроводник (КМОП)
CPU Центральный процессор
CS Chip Select или Crystall Select (тоже CE)
E2PROM Electrically Erasable PrograMMable ROM
EEPROM Electrically Erasable PrograMMable ROM
EPROM Erasable PrograMMable ROM
FIFO Структура данных, организованная по принципу "первым вошел - первым вышел" (конвейер или очередь)
FRAM Ferroelectric RAM
LED Светодиод
LIFO Структура данных, организованная по принципу "последним вошел - первым вышел" (стек)
MRAM Magnitoelectric RAM
OTP One TiMe PrograMMable (PROM)
OUM Ovonic Unified MeMory
PROM PrograMMable ROM (OTP)
RAM RandoM Access MeMory (ОЗУ, ЗУПВ)
ROM Read Only MeMory (ПЗУ)
АЛУ Арифметико-логическое устройство (ALU)
АЦП Аналого-цифровой преобразователь
БИС Большая интегральная схема
ВВ Ввод - вывод
ВУ Внешнее устройство
ЖКД Жидко-кристаллический дисплей (LCD)
ЖКИ Жидко-кристаллический индикатор

ЗУПВ Запоминающее устройство с произвольной выборкой (ОЗУ)
ИМС Интеральная микросхема (тоже ИС)
ИС Интегральная схема (тоже ИМС)
КЛС Комбинационная логическая схема (тоже КС)
КМОП К-МОП комплементарная логика на транзисторах металл-оксид-полупроводник (CMOS)
КОП Код операции
КС Это КЛС
ЛБ Логический базис
ЛФ Логическая функция
ЛЭ Логический элемент
МК Микроконтроллер
МП Микропроцессор
ОЗУ Оперативное запоминающее устройство (ЗУПВ)
ОК Открытый коллектор
ОЭВМ Однокристальная ЭВМ (микроконтроллер)
ПДП Прямой доступ к памяти (DMA)
ПЗУ Постоянное запоминающее устройство (ROM)
ПИТ Программируемый нтервальный таймер
ППИ Программируемый периферийный интерфейс
ПС Последовательностная схема
ПФ Переключательная функция
РОН Регистр общего назначения
САПР Система автоматизированного проектирования
СБИС Сверхбольшая интегральная схема
СДНФ Совершенная дизъюнктивная нормальная форма
ТИ Таблица истинности
ТК Таблица Карно
ЦАП Цифро-аналоговый преобразоватеь
ЦУ Цифровое устройство
ШИМ Широтно-импульсная модуляция (PWM)
ЭВМ Ну, это и так понятно
ЭП Элемент памяти
ЯП Ячейка памяти

СПИСОК ЛИТЕРАТУРЫ

1. Ю.В. Китаев Основы цифровой техники / Учебное пособие. - СПб: СПбГУ ИТМО, 2007
2. Китаев Ю.В. Основы программирования микроконтроллеров atmega128 и 68hc908 / Учебное пособие.: СПбГУ ИТМО, 2007.
3. Китаев Ю.В. Цифровые и микропроцессорные устройства. Учебник и задачник. WWW-адрес (http://faculty.ifmo.ru/electron), 2003.
4. Китаев Ю.В. Дистанционные лабораторные и практические работы. WWW-адрес (http://faculty.ifmo.ru/electron), 2003.
5. Угрюмов Е. П. Цифровая схемотехника. СПб., БХВ-Петербург, 2004.
6. Соловьев В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. М., Горячая линия-Телеком, 2001.
7. Бродин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. Изд-во "ЭКОМ", М., 2002г.
8. Пухальский Г.И., Новосельцева Т.Я. Цифровые устройства. СПб, Политехника, 1996.
9. Потемкин И.С. Функциональные узлы цифровой автоматики. Изд-во "Энергоатомиздат", М., 1988.
10. Опадчий Ю.Ф. и др. Аналоговая и цифровая электроника. Учебник для вузов. М.: Горячая линия-Телеком, 1999.
11. Антонов А.П. Язык описания цифровых устройств. Изд-во "РадиоСофт", М., 2002.
12. Поляков А.К. Языки VHDL и Verylog в проектировании цифровой аппаратуры. Изд-во "СОЛОН-пресс", М., 2003.
13. Новиков Ю.В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. М., Мир, 2001.
14. Новожилов О.П. Основы цифровой техники. М., РадиоСофт, 2004.

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

КАФЕДРА ЭЛЕКТРОНИКИ

Заведующий кафедрой: д.т.н., проф. Г.Н. Лукьянов.

Кафедра Электроники (первоначальное название “Радиотехники”) была основана в 1945 году. Первым руководителем кафедры был С.И. Зилитинкевич известный в стране и за рубежом ученый в области физической электроники и радиотехники, активный работник высшей школы, заслуженный деятель науки и техники РСФСР, доктор технических наук, профессор ЛИТМО с 1938 г., инициатор создания в ЛИТМО инженерно-физического и радиотехнического факультетов (1946г.). С.И. Зилитинкевич заведовал кафедрой с 1945 до 1978 года. Под его научным руководством аспирантами и соискателями выполнено более 50 кандидатских диссертаций, многие его ученики стали докторами наук.

В дальнейшем, с 1978 г. по 1985 г. кафедру возглавил к.т.н., доцент Е.К. Алахов, один из учеников С.И. Зилитинкевича.

С 1985 г. по 2006 г. руководителем кафедры стал д.т.н., профессор В.В. Тогатов, известный специалист в области силовой электроники и приборов для измерения параметров полупроводниковых структур.

Начиная с 2006 г. кафедрой заведует д.т.н., профессор Г.Н. Лукьянов, под руководством и при участии которого кардинально обновилось лабораторное оборудование в рамках инновационной программы развития.

Основные направления кафедры связаны с разработкой приборов для лазерной и медицинской техники, приборов для измерения параметров полупроводниковых структур, а также встраиваемых цифровых и микропроцессорных устройств.

Под руководством В.В. Тогатова было разработано и изготовлено большое число приборов различного назначения:

- Измеритель параметров ультрабыстрых диодов;

- Универсальное устройство для исследования переходных процессов в силовых полупроводниковых структурах;
- Измеритель времени жизни заряда в слаболегированных областях диодных, тиристорных и транзисторных структур;
- Универсальный разрядный модуль для накачки твердотельных лазеров;
- Импульсный источник токов для накачки лазерных линеек;
- Высокочастотный разрядный модуль для систем накачки твердотельных лазеров и импульсных источников света;
- Программируемый источник света для питания галогенных ламп;
- Блок управления затвором с нарушением полного внутреннего отражения;
- и много других.

На кафедре написаны и размещены на сайте ЦДО следующие материалы для дистанционного обучения (автор Ю.В. Китаев):

- Конспект лекций по дисциплине “Электроника и микропроцессорная техника”;
- свыше 600 вопросов к обучающим и аттестующим тестам;
- 18 дистанционных лабораторных и практических работ

На кафедре имеются следующие компьютеризированные учебные лаборатории:

- АРМС – полупроводниковые приборы;
- Устройства на полупроводниковых приборах;
- Цифровая техника;
- Микропроцессорная техника
- Моделирование электронных устройств.

Юрий Васильевич Китаев

Лабораторные и практические работы “Электроника и МП техника” часть 1

Учебное пособие

В авторской редакции
Дизайн и верстка Ю.В. Китаев
Редакционно-издательский отдел Санкт-Петербургского государственного университета информационных технологий, механики и оптики
Зав. РИО Н.Ф. Гусарова
Лицензия ИД 3 00408 от 05.11.99
Подписано к печати _____ Тираж 100 экз.
Заказ № 1371 Отпечатано на ризографе