МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт высокоточных систем им. В.П. Грязева

Кафедра систем автоматического управления

Утверждено на заседании кафедры
«Системы автоматического управления»
«22» января 2020 г., протокол № 6

Заведующий кафедрой

_____О.В.Горячев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине (модулю)

«Микропроцессорные устройства систем управления движением летательных аппаратов»

основной профессиональной образовательной программы
высшего образования – программы специалитета

по специальности
24.05.06 Системы управления летательными аппаратами

со специализацией
Системы управления движением летательных аппаратов

Форма обучения: очная

Идентификационный номер образовательной программы: 240506-01-20

Тула 2020 год

Разработчик(и) методических указаний

Морозов О.О., доцент, к.т.н. _____

(ФИО, должность, ученая степень, ученое звание) (подпись)

Лабораторная работа № 1. Написание управляющих программ на ассемблере для микропроцессоров семейства x86 и микроконтроллера МК51.

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение системы команд микропроцессоров семейства X86 и микро-ЭВМ i8051, написание программ на ассемблере.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Система команд микро-ЭВМ включает в свой состав 111 основных команд. Длина команд составляет 1, 2 или 3 байта, причем большинство команд (94%) одно- или двухбайтовые. Все команды выполняются за 1 или 2 машинных цикла (1.0 или 2.0мкс при тактовой частоте 12МГц соответственно), исключение составляют лишь команды умножения и деления, которые выполняются за 4 машинных цикла (4.0мкс).

В качестве операндов команд микро-ЭВМ может использовать: отдельные биты, 4-х битные цифры, байты и двухбайтовые слова. Всего микро-ЭВМ выполняет 13 типов команд (рис. 1).

Тип 1	D7	D6	D5	D4	D3	D2	D1	D0								
	КОП															
Тип 2	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	КОП								#d							
Тип 3	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	КОП								ad							
Тип 4	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	КОП								Bit							
Тип 5	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	КОП								rel							
Тип 6	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
	A10	A9	A8	КОП					A7	A6	A5	A4	A3	A2	A1	A0
Тип 7	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								ad				#d			
Тип 8	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								ad				Rel			
Тип 9	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								ad				Add			
Тип 10	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								#d				rel			
Тип 11	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								Bit				rel			
Тип 12	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								Ad16h				A16l			
Тип 13	D7 ... D0								D7 ... D0				D7 ... D0			
	КОП								D16h				D16l			

Рис 1. Типы команд микро-ЭВМ.

Как следует из рис.1 первый байт команды всегда содержит код операции (КОП), а второй и третий байты (если они присутствуют в команде) содержат адреса операндов или непосредственные значения операндов.

Наиболее существенным отличием системы команд рассматриваемой микро-ЭВМ от микро-ЭВМ семейства BE48, является возможность адресовать отдельные биты в резидентной памяти данных. Кроме того, выше уже отмечалось, что отдельные регистры из блока регистров специальных функций так же допускают адресацию отдельных бит.

Карты адресов отдельных бит в резидентной памяти данных и в блоке регистров специальных функций приведены на рис 2 и 3.

7Fh 30h	. . .							
2Fh	7F	7E	7D	7C	7B	7A	79	78
2Eh	77	76	75	74	73	72	71	70
2Dh	6F	6E	6D	6C	6B	6A	69	68
2Ch	67	66	65	64	63	62	61	60
2Bh	5F	5E	5D	5C	5B	5A	59	58
2Ah	57	56	55	54	53	52	51	50
29h	4F	4E	4D	4C	4B	4A	49	48
28h	47	46	45	44	43	42	41	40
27h	3F	3E	3D	3C	3B	3A	39	38
26h	37	36	35	34	33	32	31	30
25h	2F	2E	2D	2C	2B	2A	29	28
24h	27	26	25	24	23	22	21	20
23h	1F	1E	1D	1C	1B	1A	19	18
22h	17	16	15	14	13	12	11	10
21h	0F	0E	0D	0C	0B	0A	09	08
20h	07	06	05	04	03	02	01	00
1Fh 18h	Банк 3							
17h 10h	Банк 2							
0Fh 08h	Банк 1							
07h 00h	Банк 0							

Рис 2. Карта адресуемых бит в резидентной памяти данных.

1Fh	. . .								
F0h	F7	F6	F5	F4	F3	F2	F1	F0	B
	. . .								
E0h	E7	E6	E5	E4	E3	E2	E1	E0	A
	. . .								
D0h	D7	D6	D5	D4	D3	D2	D1	D0	PSW
	. . .								
B8h	-	-	-	BC	BB	BA	B9	B8	IP
	. . .								
B0h	B7	B6	B5	B4	B3	B2	B1	B0	P3
	. . .								
A8h	AF	-	-	AC	AB	AA	A9	A8	IE
	. . .								
A0h	A7	A6	A5	A4	A3	A2	A1	A0	P2
	. . .								
98h	9F	9E	9D	9C	9B	9A	99	98	SCON
	. . .								
90h	97	96	95	94	93	92	91	90	P1
	. . .								
88h	8F	8E	8D	8C	8B	8A	89	88	TCON
	. . .								
80h	87	86	85	84	83	82	81	80	P0

Рис 3. Карта адресуемых бит в блоке регистров специальных функций.

Все множество команд микро-ЭВМ можно разбить на 5 функциональных групп:

- команды пересылки данных;
- команды арифметических операций;
- команды логических операций;
- команды операций над битами;
- команды передачи управления.

При рассмотрении всех команд будут использоваться следующие обозначения:

Rn (n=0,1,...,7) - регистр общего назначения в выбранном банке регистров;

@Ri (i=0,1) - регистр общего назначения в выбранном банке регистров, используемый в качестве регистра косвенного адреса;

ad - адрес прямоадресуемого байта;

ads - адрес прямоадресуемого байта-источника;

add - адрес прямоадресуемого байта-получателя;

ad11 - 11-разрядный абсолютный адрес перехода;

ad16 - 16-разрядный абсолютный адрес перехода;

rel - относительный адрес перехода;

#d - непосредственный операнд;

#d16 - непосредственный операнд (2 байта);

bit - адрес прямоадресуемого бита;

/bit - инверсия прямоадресуемого бита;

A - аккумулятор;

PC - счетчик команд;

DPTR - регистр указатель данных;

( ) - содержимое ячейки памяти или регистра.

Группа команд пересылки данных.

Данная группа команд представлена 28 командами, краткое описание которых приведено в таблице 1, где так же указаны тип команды (T) в соответствии с рисунком 1, длина команды в байтах (B) и время выполнения команды в машинных циклах (C).

Таблица 1.

Команды пересылки данных.

Мнемокод	КОП	T	B	C	Описание
MOV A,Rn	11101rrr	1	1	1	(A) <-- (Rn)
MOV A,ad	11100101	3	2	1	(A) <-- (ad)
MOV A,@Ri	1110011i	1	1	1	(A) <-- ((Ri))
MOV A,#d	01110100	2	2	1	(A) <-- #d
MOV Rn,A	11111rrr	1	1	1	(Rn) <-- (A)
MOV Rn,ad	10101rrr	3	2	2	(Rn) <-- (ad)
MOV Rn,#d	01111rrr	2	2	1	(Rn) <-- #d
MOV ad,A	11110101	3	2	1	(ad) <-- (A)
MOV ad,Rn	10001rrr	3	2	2	(ad) <-- (Rn)
MOV add,ads	10000101	9	3	2	(add) <-- (ads)
MOV ad,@Ri	1000011i	3	2	2	(ad) <-- ((Ri))
MOV ad,#d	01110101	7	3	2	(ad) <-- #d
MOV @Ri,A	1111011i	1	1	1	((Ri)) <-- (A)
MOV @Ri,ad	0110011i	3	2	2	((Ri)) <-- (ad)
MOV @Ri,#d	0111011i	2	2	1	((Ri)) <-- #d
MOV DPTR,!#d16	10010000	13	3	2	(DPTR) <-- #d16
MOVC A,@A+ DPTR	10010011	1	1	2	(A) <-- ((A)+(DPTR))
MOVC A,@A+PC	10000011	1	1	2	(PC) <-- (PC)+1) (A) <-- ((A)+(PC))
MOVX A,@Ri	1110001i	1	1	2	(A) <-- ((Ri))
MOVX A,@DPTR	11100000	1	1	2	(A) <-- ((DPTR))
MOVX @Ri,A	1111001i	1	1	2	((Ri)) <-- (A)
MOVX @DPTR,A	11110000	1	1	2	((DPTR)) <-- (A)
PUSH ad	11000000	3	2	2	(SP) <-- (SP)+1 ((SP)) <-- (ad)
POP ad	11010000	3	2	2	(ad) <-- ((SP)) (SP) <-- (SP)-1
XCH A,Rn	11001rrr	1	1	1	(A) <-> (Rn)
XCH A,ad	11000101	3	2	1	(A) <-> (ad)
XCH A,@Ri	1100011i	1	1	1	(A) <-> ((Ri))
XCHD A,@Ri	1101011i	1	1	1	(A0-3) <-> ((Ri0-3))

По команде MOV выполняется пересылка данных из второго операнда в первый. Команда MOV не имеет доступа ни к внешней памяти данных, ни к памяти программ. Для этих целей служат команды MOVX и MOVC

соответственно. Команда MOVX обеспечивает чтение/запись байт из внешней памяти данных, а команды MOVC - чтение байт из памяти программ.

По команде XCH выполняется обмен байтами между аккумулятором и ячейкой РПД, а по команде XCHD - обмен младшими тетрадами (битами 0-3).

Если сравнить группы команд пересылки данных рассматриваемой микро-ЭВМ и микро-ЭВМ семейства BE48, то можно выделить два существенных отличия. Во-первых, в рассматриваемой микро-ЭВМ "исчезли" все команды работа со специальными регистрами: PSW, таймером, портами ввода/вывода. Теперь доступ к ним, как и к другим регистрам специальных функций осуществляется путем задания соответствующего прямого адреса, т.е. эти команды упрятаны в команды типа 3. Например, чтение PSW в аккумулятор может быть выполнено командой

MOV A,PSW ,

которая преобразуется ассемблером к виду

MOV A,0D0h (E5 D0) ,

где E5 - код операции, а D0 - операнд (адрес PSW).

Здесь следует отметить, что большинство ассемблеров допускают символические имена для регистров специальных функций, а отдельные биты этих регистров (конечно если выбранный регистр допускает адресацию отдельных бит) могут адресоваться путем указания имени регистра и номера бита через точку. Например, к нулевому биту аккумулятора можно обратиться по имени ACC.0. Это означает, что в микро-ЭВМ аккумулятор имеет два различных имени в зависимости от способа адресации: A - при неявной адресации (например, MOV A,R0) и ACC - при использовании прямого адреса. Первый способ предпочтительнее, однако не всегда возможный.

Вторым существенным отличием является появление команд записи данных в стек PUSH и их чтения из стека POP. Размер стека теперь ограничен лишь размером резидентной памяти данных.

Группа команд арифметических операций.

Данная группа команд состоит из 24 команд, краткое описание которых приведено в таблице 2.

Таблица 2.

Команды арифметических операций.

Мнемокод	КОП	T	B	C	Описание
ADD A,Rn	00101rrr	1	1	1	(A) <-- (A)+(Rn)
ADD A,ad	00100101	3	2	1	(A) <-- (A)+(ad)
ADD A,@Ri	0010011i	1	1	1	(A) <-- (A)+((Ri))
ADD A,#d	00100100	2	2	1	(A) <-- (A)+#d
ADDC A,Rn	00111rrr	1	1	1	(A) <-- (A)+(Rn)+(C)
ADDC A,ad	00110101	3	2	1	(A) <-- (A)+(ad)+(C)
ADDC A,@Ri	0011011i	1	1	1	(A) <-- (A)+((Ri))+(C)
ADDC A,#d	00110100	2	2	1	(A) <-- (A)+#d+(C)
DA A	11010100	1	1	1	десятичная коррекция аккумулятора
SUBB A,Rn	10011rrr	1	1	1	(A) <-- (A)-(Rn)-(C)
SUBB A,ad	10010101	3	2	1	(A) <-- (A)-(ad)-(C)
SUBB A,@Ri	1001011i	1	1	1	(A) <-- (A)-((Ri))-(C)
SUBB A,#d	10010100	2	2	1	(A) <-- (A)-#d-(C)
INC A	00000100	1	1	1	(A) <-- (A)+1
INC Rn	00001rrr	1	1	1	(Rn) <-- (Rn)+1
INC ad	00000101	3	2	1	(ad) <-- (ad)+1
INC @Ri	0000011i	1	1	1	((Ri)) <-- ((Ri))+1
INC DPTR	10100011	1	1	2	(DPTR) <-- (DPTR)+1
DEC A	00010100	1	1	1	(A) <-- (A)-1

DEC Rn	00011rrr	1	1	1	(Rn) <-- (Rn)-1
DEC ad	00010101	3	2	1	(ad) <-- (ad)-1
DEC @Ri	0001011i	1	1	1	((Ri)) <-- ((Ri))-1
MUL AB	10100100	1	1	4	(B)(A) <-- (A)*(B)
DIV AB	10000100	1	1	4	(A).(B) <-- (A)/(B)

Как следует из таблицы 2 следует, что микро-ЭВМ выполняет достаточно широкий набор команд для организации обработки целочисленных данных, включая команды умножения и деления. По результату выполнения команд ADD, ADDC, SUBB, MUL и DIV устанавливаются флаги PSW, структура которого приведена на рис.4.

D7	D6	D5	D4	D3	D2	D1	D0
C	AC	F0	RS1	RS0	OV	-	P
признак переноса	признак дополни-тельного переноса	флаг пользователя 0	селектор банка регистров		признак переполнения		признак паритета

Рис 4. Формат PSW.

Флаг C устанавливается при переносе из разряда D7, т.е. когда результат не помещается в 8 разрядов.

Флаг AC устанавливается при переносе из разряда D3 в командах ссложения и вычитания и служит для реализации десятичной арифметики. Этот признак используется командой DA A.

Флаг OV устанавливается при переносе из разряда D6, т.е. когда результат не помещается в 7 разрядов и восьмой разряд не может быть интерпретирован как знаковый. Этот признак служит для организации обработки чисел со знаком.

Флаг P устанавливается и сбрасывается аппаратно. Если число единичных бит в аккумуляторе нечетно, то P=1, иначе P=0.

Группа команд логических операций.

Данная группа команд состоит из 25 команд, краткое описание которых приведено в таблице 3.

Таблица 3.

Команды логических операций.

Мнемокод	КОП	T	B	C	Описание
ANL A,Rn	01011rrr	1	1	1	(A) <-- (A)/\(Rn)
ANL A,ad	01010101	3	2	1	(A) <-- (A)/\(ad)
ANL A,@Ri	0101011i	1	1	1	(A) <-- (A)/\((Ri))
ANL A,#d	01010100	2	2	1	(A) <-- (A)/\#d
ANL ad,A	01010010	3	2	1	(ad) <-- (ad)/\(A)
ANL ad,#d	01010011	7	3	2	(ad) <-- (ad)/\#d
ORL A,Rn	01001rrr	1	1	1	(A) <-- (A)\/(Rn)
ORL A,ad	01000101	3	2	1	(A) <-- (A)\/(ad)
ORL A,@Ri	0100011i	1	1	1	(A) <-- (A)\/((Ri))
ORL A,#d	01000100	2	2	1	(A) <-- (A)\/#d
ORL ad,A	01000010	3	2	1	(ad) <-- (ad)\/A
ORL ad,#d	01000011	7	3	2	(ad) <-- (ad)\/#d
XRL A,Rn	01101rrr	1	1	1	(A) <-- (A) (+) (Rn)
XRL A,ad	01100101	3	2	1	(A) <-- (A) (+) (ad)
XRL A,@Ri	0110011i	1	1	1	(A) <-- (A) (+) ((Ri))
XRL A,#d	01100100	2	2	1	(A) <-- (A) (+) #d
XRL ad,A	01100010	3	2	1	(ad) <-- (ad) (+) A
XRL ad,#d	01100011	7	3	2	(ad) <-- (ad) (+) #d
CLR A	11100100	1	1	1	(A) <-- 0
CPL A	11110100	1	1	1	(A) <-- NOT (A)
SWAP A	11000100	1	1	1	(A0-3) <-> (A4-7)
RL A	00100011	1	1	1	циклический сдвиг влево
RLC A	00110011	1	1	1	сдвиг влево через перенос
RR A	00000011	1	1	1	циклический сдвиг вправо
RRC A	00010011	1	1	1	сдвиг вправо через перенос

Как следует из таблицы 3, данная группа команд позволяет выполнять операции над байтами: логическое И (/\ ), логическое ИЛИ (\/), исключающее ИЛИ ((+)), инверсию (NOT), сброс в нулевое значение и сдвиг. Команды оперирующие отдельными битами описаны далее.

Группа команд операций над битами.

Данная группа команд состоит из 12 команд, краткое описание которых приведено в таблице 4.

Таблица 4.

Команды операций над битами.

Мнемокод	КОП	T	B	C	Описание
CLR C	11000011	1	1	1	(C) <-- 0
CLR bit	11000010	4	2	1	(bit) <-- 0
SETB C	11010011	1	1	1	(C) <-- 1
SETB bit	11010010	4	2	1	(bit) <-- 1
CPL C	10110011	1	1	1	(C) <-- NOT (C)
CPL bit	10110010	4	2	1	(bit) <-- NOT (bit)
ANL C,bit	10000010	4	2	2	(C) <-- (C)/\(bit)
ANL C,/bit	10110000	4	2	2	(C) <-- (C)/\NOT(bit)
ORL C,bit	01110010	4	2	2	(C) <-- (C)\/(bit)
ORL C,/bit	10100000	4	2	2	(C) <-- (C)\/NOT(bit)
MOV C,bit	10100010	4	2	1	(C) <-- (bit)
MOV bit,C	10010010	4	2	2	(bit) <-- (C)

Как следует из таблицы 4, данная группа команд позволяет выполнять операции над отдельными битами: сброс, установку, инверсию бита, а так же логическое И (/\ ) и логическое ИЛИ (\/).

В качестве "логического" аккумулятора, участвующего во всех операциях с двумя операндами, выступает признак переноса C (разряд D7 PSW). В качестве операндов могут использоваться 128 бит из резидентной памяти данных и регистры специальных функций допускающие адресацию отдельных бит.

Группа команд передачи управления.

Группа команд передачи управления представлена командами без условного и условного переходов, командами вызова подпрограмм и командами возврата из подпрограмм. Краткое описание команд группы приведено в таблице 5.

Таблица 5.

Команды передачи управления.

Мнемокод	КОП	T	B	C	Описание
LJMP ad16	00000010	12	3	2	длинный безусловный переход по всей памяти
AJMP ad11	a10 a9 a8 00001	6	2	2	безусловный переход в пределах страницы 2 Кбайт
SJMP rel	10000000	5	2	2	безусловный переход в пределах страницы 256 байт
JMP @A+DPTR	01110011	1	1	2	безусловный переход по косвенному адресу
JZ rel	01100000	5	2	2	переход если нуль
JNZ rel	01110000	5	2	2	переход если не нуль
JC rel	01000000	5	2	2	переход если бит переноса установлен
JNC rel	01010000	5	2	2	переход если бит переноса не установлен
JB bit,rel	00100000	11	3	2	переход если бит установлен
JNB bit,rel	00110000	11	3	2	переход если бит не установлен
JBC bit,rel	00010000	11	3	2	переход если бит установлен со сбросом бита
DJNZ Rn,rel	11011rrr	5	2	2	команда цикла
DJNZ ad,rel	11010101	8	3	2	команда цикла
CJNE A,ad,rel	10110101	8	3	2	Сравнение аккумулятора с байтом и переход если не равно
CJNE A,#d,rel	10110100	10	3	2	сравнение аккумулятора с константой и переход если не равно

CJNE Rn,#d,rel	10111rrr	10	3	2	сравнение регистра с константой и переход если не равно
CJNE @Ri,#d,rel	1011011i	10	3	2	сравнение байта памяти с константой и переход если не равно
LCALL ad16	00010010	12	3	2	длинный вызов подпрограммы во всей памяти
ACALL ad11	a10 a9 a8 10001	6	2	2	вызов подпрограммы в пределах страницы 2 Кбайт
RET	00100010	1	1	2	возврат подпрограммы
RETI	00110010	1	1	2	возврат подпрограммы обработки прерывания
NOP	00000000	1	1	1	пустая операция

Команда безусловного перехода LJMP (L-long-длинный) осуществляет переход по абсолютному 16-битному адресу, указанному в теле команды, т.е. команда обеспечивает переход в любую точку памяти программ. Действие команды AJMP (A-absolute-абсолютный) аналогично команде LJMP, однако в теле команды указаны лишь 11 младших разрядов адреса. Поэтому переход осуществляется в пределах страницы размером в 2 Кб, при этом надо иметь в виду, что сначала содержимое счетчика команд увеличивается на 2 и только потом заменяются младшие 11 разрядов адреса.

В отличие от предыдущих в команде SJMP (S-short-короткий) указан не абсолютный, а относительный адрес перехода. Величина смещения rel рассматривается как число со знаком, а следовательно переход возможен в пределах -128...+127 байт относительно адреса команды следующей за командой SJMP.

Команда косвенного перехода JMP @A+DPTR позволяет вычислять адрес перехода в процессе выполнения самой программы.

Команды условного перехода позволяют проверять следующие условия:

JZ - аккумулятор содержит нулевое значение;

JNZ - аккумулятор содержит не нулевое значение;

JC - бит переноса C установлен;

JNC - бит переноса C не установлен;

JB - прямоадресуемый бит равен единице;

JNB - прямоадресуемый бит равен нулю;

JBC - прямоадресуемый бит равен единице и сбрасывается в нулевое значение при выполнении команды.

В отличие от микро-ЭВМ семейства BE48, все команды условного перехода содержат короткий относительный адрес, т.е. переход может осуществляться в пределах -128...+127 байт относительно следующей команды.

Команда DJNZ предназначена для организации программных циклов. Регистр Rn или байт по адресу ad, указанные в теле команды, содержат счетчик повторений цикла, а смещение rel – относительный адрес перехода к началу цикла. При выполнении команды содержимое счетчика уменьшается на единицу и проверяется на нуль. Если значение счетчика не равно нулю, то осуществляется переход на начало цикла, иначе выполняется следующая команда.

Команда CJNZ удобна для реализации процедур ожидания внешних событий. В теле команды указаны "координаты" двух байт и относительный адрес перехода rel. В качестве двух байт могут быть, например, использованы значения аккумулятора и прямоадресуемого байта или косвенно адресуемого байта и константы. При выполнении команды значения указанных двух байт

сравниваются и, в случае, если эти значения не равны, осуществляется переход. Например, команда

WAIT: CJNE A,P0,WAIT

будет выполняться до тех пор, пока значения на линиях порта P0 не совпадут с аккумулятором.

Действие команд вызова процедур полностью аналогично командам безусловного перехода. Единственное отличие состоит в том, что эти команды сохраняют в стеке адрес возврата.

Команда возврата из подпрограммы RET восстанавливает из стека значение счетчика команд, а команда возврата из процедуры обработки прерывания RETI, кроме того разрешает прерывание обслуженного уровня. Команды RET и RETI не различают какой командой LCALL или ACALL подпрограмма была вызвана, т.к. и в том и в другом случае в стеке сохраняется полный 16 разрядный адрес возврата.

В заключении следует отметить, что большинство ассемблеров допускают обобщенную мнемонику JMP для команд безусловного перехода и CALL для команд вызова подпрограмм. Конкретный тип команды определяется ассемблером исходя из степени "длины" перехода или вызова.

Директивы языка

Директивы выполняются ассемблером в процессе преобразования исходного модуля в объектный. В языке ассемблера имеются следующие виды директив:

1) определения названия программного модуля NAME;
2) определения имен перемещаемых сегментов SEGMENT;
3) определения перемещаемых сегментов RSEG;
4) определения абсолютных сегментов CSEG, XSEG, DSEG, ISEG и BSEG;
5) управления счетчиком адреса ORG;
6) определения данных в пространстве памяти программ DB и DW;
7) резервирования памяти DS и DBIT;
8) определения имен-синонимов EQU и SET;
9) определения имен внутренних адресов CODE, XDATA, DATA, IDATA и BIT;
10) определения имен связи программных модулей PUBLIC и EXTRN;
11) управления банками рабочих регистров USING;
12) окончания программного модуля END.

Обобщенная форма записи директив:

-<имя>- <дир> -<арг>-,

где <имя> - поле имени;
<дир> - поле мнемокода директивы;
<арг> - поле аргументов.

В поле мнемокода записывается мнемоническое обозначение директивы языка. В поле имени записывается идентификатор, определяемый пользователем. В поле аргументов записывается один или несколько аргументов, которые определяют дополнительную информацию, требуемую для выполнения директивы. Разделителем поля имени и поля мнемокода, а также поля мнемокода и поля аргументов является символ интервала ("пробел" или "горизонтальная табуляция"). Разделителем аргументов является знак "запятая" или символ интервала в зависимости от вида директивы.

Для всех директив, за исключением директив определения данных DB, DW и резервирования памяти DS, DBIT в обобщенной форме записи предложения, поле метки запрещено.

1. Директива определения названия программного модуля NAME

Форма записи директивы:

NAME <назв> ,

Где <назв> - название.

Данная директива присваивает текущему программному модулю название <имя>. В исходном модуле может быть записана только одна директива NAME, причем ей могут предшествовать только предложения, содержащие комментарий к программе или пустые предложения. Название программного модуля <имя> может использоваться в качестве идентификатора в этом же исходном модуле. Правила записи названия модуля такие же, как и для идентификаторов, за исключением:

1) максимальное количество символов в названии 40;

2) в качестве буквы в названии разрешается использовать знак "коммерческое эт" (@).

Примеры использования директивы NAME:

NAME синус

NAME @PTR

NAME PROCEDURE_51

2. Директива определения имен перемещаемых сегментов SEGMENT

Форма записи директивы:

<имя_сег> SEGMENT <тип_сег> -<тип_перем>- ,

где <имя_сег> - идентификатор, представляющий имя перемещаемого сегмента;

<тип_сег> - тип сегмента, представленный вспомогательным словом CODE, XDATA, DATA, IDATA или BIT;

<тип_перем> - тип перемещения сегмента, представленный вспомогательным словом PAGE, INPAGE, INBLOCK, BITADDRESSABLE или UNIT.

Данная директива определяет имя перемещаемого сегмента и связанные с ним атрибуты сегмента: тип сегмента, тип перемещения сегмента и величину базового адреса сегмента. Алгоритм установки атрибутов, связанных с именем сегмента:

нач
тип(имя_сег) := адрес
сег(имя_сег) := тип_сег
фикс(имя_сег):= перем
вел(имя_сег) := неопред
кон

Имя перемещаемого сегмента может быть объявлено в исходном модуле только один раз. Это имя может использоваться в директиве RSEG, при создании перемещаемого сегмента и в качестве терма выражения.

Тип сегмента определяет тип пространства памяти, в котором должен размещаться сегмент:

Тип сегмента	Тип пространства памяти
CODE	пространство памяти программ
XDATA	пространство внешней памяти данных
DATA	пространство внутренней непосредственно адресуемой памяти данных
IDATA	пространство внутренней косвенно адресуемой памяти данных
BIT	пространство внутренней побитово адресуемой памяти данных

Тип перемещения сегмента определяет способ размещения именуемого сегмента в соответствующем пространстве памяти следующим образом:

Тип перемещения	Способ размещения
PAGE	на границу страницы памяти (т.е. с адреса, кратного 256)
INPAGE	внутри страницы памяти (размер страницы – 256 байт, адрес страницы кратен 256)
INBLOCK	внутри блока памяти(размер блока - 2048 байт, адрес блока кратен 2048)
BITADDRESSABLE	внутри параграфа памяти (размер параграфа - 16 байт, адрес параграфа кратен 16)
UNIT, "пусто"	внутри соответствующего пространства памяти

Допустимые наборы атрибутов "тип сегмента" и "тип перемещения сегмента":

Тип сегмента	Тип перемещения сегмента				
	PAGE	INPAGE	INBLOCK	BITADDRESSABLE	UNIT,"пусто"
CODE	+	+	+	-	+
XDATE	+	+	-	-	+
DATA	-	-	-	+	+
IDATA	-	-	-	-	+
BIT	-	-	-	-	+

Пример использования директивы для определения сегмента стека:

```
STACK   SEGMENT IDATA
        RSEG    STACK
        DS   10H   ; пространство стека
TOS     EQU  $     ; верхушка стека
        . . .
        MOV  SP,#TOS ;инициализация указателя стека
```

3. Директива определения перемещаемых сегментов RSEG

Форма записи директивы:

RSEG <имя_сег> ,

где <имя_сег> - имя перемещаемого сегмента, предварительно объявленное директивой SEGMENT.

По данной директиве следующие за ней конструкции языка в соответствии с таблицей используются для определения или управления содержимым этого перемещаемого сегмента. Атрибуты сегмента устанавливаются по предварительно объявленному директивой SEGMENT имени сегмента <имя_сег>.

В программном модуле может размещаться несколько директив RSEG с одинаковым именем сегмента. В этом случае только первая директива устанавливает счетчик адреса сегмента в нулевое состояние, последующие директивы восстанавливают конечное состояние счетчика адреса предыдущей части этого сегмента.

Конструкция языка	CODE	XDATA	DATA	IDATA	BIT
Метка	+	+	+	+	+
Инструкция	+	-	-	-	-
Директива ORG	+	+	+	+	+
Директива DB и DW	+	-	-	-	-
Директива DS	+	+	+	+	-
Директива DBIT	-	-	-	-	+

Пример использования директивы RSEG:

```
FIRST SEGMENT DATA      ; определение имени перемещаемого
                        ; сегмента типа DATA
SECOND SEGMENT CODE     ; определение имени перемещаемого
                        ; сегмента типа CODE
RSEG FIRST              ; определение сегмента FIRST TOTAL : DS
1
COUNT1: DS 1
COUNT2: DW 2
RSEG SECOND             ; определение сегмента SECOND CONST: DB
'01234'
START: MOV A,#CONST
```

4. Директивы определения абсолютных сегментов CSEG, XSEG, DSEG, ISEG и BSEG

Форма записи директив:

CSEG=XSEG=DSEG=ISEG=BSEG –AT <абс_выр>- ,

где <абс_выр> - абсолютное выражение, не содержащее послеопределенных термов.

По данной директиве следующие за ней инструкции языка в соответствии с табл. 4 используются для определения или управления содержимым абсолютного сегмента. Тип определяемого сегмента устанавливается по мнемокоду директивы следующим образом:

Мнемокод директивы	Тип определяемого сегмента
CSEG	CODE
XSEG	XDATA
DSEG	DATA
ISEG	IDATA
BSEG	BIT

В программном модуле может размещаться несколько директив с одинаковыми мнемокодами. Первая директива определяет начальную часть

сегмента, последующие директивы определяют части продолжения этого же сегмента.

Конструкция AT <абс_выр> используется для установки содержимого счетчика адреса начальной части или части продолжения сегмента в состояние, равное значению выражения <абс_выр>.

Если директива определения начальной части сегмента не содержит конструкции AT <абс_выр>, то счетчик адреса сегмента устанавливается в нулевое состояние.

Если директива определения части продолжения сегмента не содержит конструкции AT <абс_выр>, то эта директива восстанавливает конечное состояние счетчика адреса, установленное предыдущей частью сегмента.

Конструкции языка, предшествующие первой директиве определения абсолютных или перемещаемых сегментов, считаются принадлежащими к абсолютному сегменту типа CODE.

Пример использования директивы BSEG и CSEG:

```
BSEG AT 70H
MODE DBIT 1
FLAG DBIT 1
      CSEG
      . . .
      MOV C,MODE
```

5. Директива управления счетчиком адреса ORG

Форма записи директивы:

ORG <выр> ,

где <выр> - абсолютное или простое переместимое выражение, несодержащее послеопределенных термов. Атрибут "сегмент" выражения должен быть или неопределенен или должен соответствовать типу текущего сегмента.

Данная директива устанавливает счетчик адреса текущего сегмента (<тек_сег>) в состояние, равное значению выражения <выр>. Доступ к значению счетчика адреса в выражении осуществляется через указатель счетчика адреса ($). Алгоритм установки атрибутов указателя счетчика адреса директивой ORG:

```
нач
тип($):= адрес
сег($):= сег(тек_сег)
фикс($):= фикс(тек_сег)
вел($):= вел(выр)
кон
```

Новое содержимое счетчика адреса не должно превышать максимального адреса пространства памяти, в котором размещается сегмент с учетом ограничения на размещение этого сегмента, определенного типом перемещения (см. Директиву SEGMENT). После трансляции очередной инструкции языка содержимое текущего счетчика адреса сегмента увеличивается на число байтов

объектного кода, генерируемых по этой инструкции. После трансляции очередного аргумента директив DB или DW содержимое текущего счетчика адреса сегмента увеличивается на число байтов объектного кода, генерируемых по этому аргументу. При определении нового сегмента состояние счетчика адреса предыдущего сегмента запоминается и создается новый счетчик адреса для определяемого сегмента со значением, равным нулю. При продолжении уже существующего сегмента содержимое счетчика адреса этого сегмента восстанавливается.

Пример использования директивы ORG:

```
ORG ($ + 10H) AND 0FFF0H
ORG 100H
```

6. Директивы определения данных в пространстве памяти программ

6.1. Директива DB

Форма записи директивы:

-<метка>:- DB <выр>=<стр>-, ... - ,

где <метка> - имя метки;

<выр> - абсолютное или общее переместимое выражение, которое может содержать послеопределенные термы со значением от 0 до 255;

<стр> - литерная строка.

Данная директива генерирует последовательность 8- битных данных (байтов данных), определенных списком аргументов, предназначенных для размещения в пространстве памяти программ. Директива выполняется следующим образом. Определяется значение выражения <выр> или код очередного символа литерной строки <стр>. Это значение рассматривается как байт данных, который последовательно включается в объектный модуль. Количество аргументов в директиве ограничено максимальным размером предложения.

Если выражение <выр> является перемещаемым, то оно должно использоваться в качестве терма оператора LOW или HIGH.

Пример использования директивы DB:

```
PRIMES: DB 1,2,3,5,7,11,13,17,19,23,29,31,37,41,43, 47, 53
MSG: DB 'ассемблер ASM51', 0DH, 0AH
```

6.2. Директива DW

Форма записи директивы:

-<метка>:- DW <выр>-, ... - ,

где <метка> - имя метки;

<выр> - абсолютное или общее переместимое выражение, которое может содержать послеопределенные термы со значением от 0 до 65535

Данная директива генерирует последовательность 16- битных данных (слов данных), определенных списком аргументов, предназначенных для размещения в пространстве памяти программ.

Директива выполняется следующим образом: определяются значения выражений <выр>. Эти значения побайтно включаются ассемблером в

объектный модуль в порядке их определения, причем старший байт значения выражения включается в объектный модуль первым.

Количество аргументов в директиве ограничено максимальным размером предложения.

Пример использования директивы DW:

```
DW   $, $+2, $+4, $+6
TABLE_PROC: DW     PROC1, PROC2, PROC3, PROC4, PROC5, PROC6
HEX: DW '01','23','45','67','89','AB', 'CD', 'EF'
```

7. Директивы резервирования памяти

7.1. Директива DS

Форма записи директивы:

-<метка>:- DS <выр> ,

где <метка> - имя метки;

<выр> - абсолютное выражение,не содержащее послеопределенных термов.

Данная директива увеличивает содержимое текущего счетчика адреса сегмента на величину, равную значению выражения <выр>.

Директива DS используется для резервирования в байтах пространства памяти программ, внешней памяти данных, непосредственно и косвенно адресуемой внутренней памяти данных.

Новое содержимое счетчика адреса не должно превышать максимального адреса соответствующего пространства памяти, в котором размещается сегмент с учетом ограничения на его размещение, указанное атрибутом "тип перемещения" текущего сегмента (см. Директиву SEGMENT).

Пример использования директивы DS:

```
        ISEG AT 5
        DS   5
I1:     DS   1
```

7.2. Директива DBIT

Форма записи директивы:

-<метка>:- DBIT <выр> ,

где <метка> - имя метки;

<выр> - абсолютное выражение, не содержащее послеопределенных термов.

Данная директива увеличивает содержимое текущего счетчика адреса сегмента на величину, равную значению выражения <выр>. Она используется для резервирования в битах пространства побитово адресуемой внутренней памяти данных. Новое содержимое счетчика адреса не должно превышать 127.

Пример использования директивы DBIT:

```
        BSEG AT 5
        DBIT 5
B1:     DBIT 1
```

8. Директивы определения имен-синонимов EQU и SET

Форма записи директив:

<имя_син> EQU=SET <выр>=<рег> ,

где <имя_син> - идентификатор, представляющий имя-синоним;

<выр> - абсолютное или простое переместимое выражение, не содержащее послеопределенных термов;

<рег> - имя программно-доступного элемента ОЭВМ.

Директива EQU или SET назначает имя-синоним выражению <выр> или имени программно-доступного элемента ОЭВМ <рег>. В первом случае имя-синоним называется именем внутреннего объекта программы типа "число" или "адрес" и может использоваться в качестве терма выражения. Алгоритм установки атрибутов терма:

нач
тип(имя_син) := тип(выр)
сег(имя_син) := сег(выр)
фикс(имя_син) := фикс(выр)
вел(имя_син) := вел(выр)
кон

В качестве термов выражения <выр> разрешается использовать другие предварительно-определенные имена-синонимы. Ссылки на послеопределенные имена-синонимы программно-доступных элементов ОЭВМ запрещены.

Имя-синоним, определенное директивой EQU, не может быть переопределено в программном модуле каким-либо способом. Имя-синоним определенное директивой SET, может быть переопределено последующей директивой SET. Кроме этого, директива SET допускает итеративное определение имени-синонима.

Пример использования директив EQU и SET:

```
ACCUM EQU A
N27 EQU 27
HERE EQU $
COUNT SET 0
      . . .
COUNT SET       COUNT + 1
```

9. Директивы определения имен внутренних адресов CODE, XDATA, DATA, IDATA и BIT

Форма записи директив:

<симв> <имя_дир> <выр>

и

<симв> BIT <сел> ,

где <имя_дир> - ключевое слово CODE, XDATA, DATA, IDATA или, BIT представляющее имя директивы;

<симв> - идентификатор, представляющий имя внутреннего адреса;

<выр> - абсолютное или простое переместимое выражение,не содержащее послеопределенных термов;

<сел> - селектор бита.

Директива <имя_дир> обьявляет идентификатор <симв>, именем адреса пространства памяти или именем адреса программно-доступного устройства ОЭВМ. Тип пространства памяти или устройства ОЭВМ определяется именем директивы следующим образом:

Имя	Тип пространства памяти или доступного устройства ОЭВМ
CODE	память программ
XDATA	внешняя память данных
DATA	внутренняя прямо адресуемая память данных, регистры специальных функций
IDATA	внутреняя косвенно адресуемая память данных
BIT	внутренняя побитово адресуемая память данных, триггеры специальных функций

Имя внутреннего адреса может использоваться в качестве терма выражения. Алгоритм установки атрибутов терма:

нач
тип(симв) := адрес
если сег(выр) = имя_дир или сег(выр) = неопред
    то сег(симв) := имя_дир
    иначе ошибка
все
фикс(симв) := фикс(выр)
вел(симв) := вел(выр=сел)
кон

Объявленный директивой идентификатор не может быть переопределен в программном модуле каким-либо способом.

Селектор бита <сел> используется только в директиве BIT для определения:

1) адреса бита в пространстве внутренней памяти данных по адресу байта и номеру бита в этом байте;

2) адреса триггера специальной функции по адресу регистра специальной функции и номеру разряда, соответствующему данному триггеру в этом регистре.

Форма записи селектора бита:

<байт>.<бит> ,

где <байт> - абсолютное или простое переместимое выражение;

<бит> - абсолютное выражение.

Абсолютное выражение <байт> определяет адрес байта в пространстве внутренней поразрядно адресуемой памяти данных или адрес поразрядно адресуемого регистра специальной функции. Атрибут "сегмент" этого выражения должен быть неопределенен или иметь значение DATA. Переместимое выражение <байт> может определять только адрес байта в пространстве внутренней поразрядно адресуемой памяти данных. Атрибут "сегмент" этого выражения должен иметь значение DATA, а тип перемещения - значение BITADDRESSABLE (см. Директиву SEGMENT).

Абсолютное выражение <бит> представляет номер бита в байте памяти или триггера в регистре специальной функции. Диапазон значения этого выражения - 0-7.

Пример использования директив:

```
INT_VEC0 CODE 03H
BUFFER IDATA 60H
BUF_LEN EQU   20
BUF_END IDATA BUFFER + BUF_LEN - 1
REL_TBL DATA REL_START + 1
CONIN       DATA SBUF
        RSEG XSEG1
        ORG  100H
DAT: DS     5
TIMEXDATA DAT + 5
        RSEG DATA_SEG CONTROL DS 1
        . . .
B_CNTRL BIT     CONTROL.0
```

10. Директивы определения имен связи программных модулей

10.1. Директива PUBLIC

Форма записи директивы:

PUBLIC <имя>-, ... - ,

где <имя> - идентификатор, представляющий определенное где-либо в программном модуле имя внутреннего объекта типа "число", имя метки или имя внутреннего объекта типа "адрес".

Данная директива объявляет идентификатор <имя> в качестве общего имени для нескольких программных модулей. На место расположения директивы в программном модуле ограничения не накладываются. Максимальное количество аргументов в директиве ограничено максимальным размером предложения языка. Один и тот же идентификатор не может быть объявлен общим именем более одного раза. Один и тот же идентификатор не может быть объявлен общим и внешним именем. Если программа формируется из нескольких модулей, то такие модули не должны содержать одинаковых общих имен.

Пример использования директивы PUBLIC:

```
PUBLIC ASC_BIN, BIN_ASC
PUBLIC размер
```

10.2. Директива EXTRN

Форма записи директивы:

EXTRN <внеш>(<ид>-, ... -) -, ... - ,

Где <внеш> - вспомогательное слово NUMBER, CODE, XDATA, DATA, IDATA или BIT;

<ид> - идентификатор, представляющий внешнее имя.

Данная директива назначает идентификатор <ид> именем внешнего объекта программы типа "число" или типа "адрес". Внешнее имя может использоваться в качестве терма выражения.

Алгоритм установки атрибутов терма:

нач
если внеш = NUMBER
    то тип(ид) := число; сег(ид) := неопред
    иначе тип(ид) := адрес; сег(ид) := внеш
все
фикс(ид):= перем
вел(ид):= неопред
кон

Пространство памяти, в котором находится объект типа "адрес", определенный внешним именем, устанавливается по элементу <внеш> директивы следующим образом:

Элемент <внеш>	Пространство памяти
CODE	память программ
XDATA	внешняя память данных
DATA	внутренняя прямо адресуемая память данных
IDATA	внутренняя косвенно-адресуемая память данных
BIT	внутренняя побитово адресуемая память данных

На объекты, объявленные внешними, можно ссылаться в текущем программном модуле. Во внешнем объектном модуле эти объекты должны быть объявлены общими. Общие и соответствующие им внешние объекты должны иметь одинаковые атрибуты "тип" и "сегмент" или один из объектов должен быть типа "число".

На размещение директивы EXTRN ограничения не накладываются. Внешнее имя не может быть переопределено в программном модуле каким-либо способом. Рекомендуется с целью оптимизации результирующего объектного кода (в частности, команд JMP и CALL) размещать директиву EXTRN до ссылки на соответствующее внешнее имя (как правило, в начале программного модуля или программного сегмента).

Пример использования директивы EXTRN:

EXTRN CODE(ASC_BIN, BIN_ASC), NUMBER(размер)

11. Директива USING

Форма записи директивы:

USING <выр> ,

где <выр> - абсолютное выражение, не содержащее послеопределенных термов со значением в пределах от 0 до 3.

Данная директива указывает на то, что в следующих инструкциях программы будет использоваться текущий банк рабочих регистров с номером, равный значению выражения <выр>. Прямой адрес для обращения к регистрам текущего банка может быть сформирован посредством ключевых слов AR0-AR7, представляющих абсолютные адреса рабочих регистров R0-R7 соответственно. Область памяти, соответствующая указанному банку рабочих регистров, резервируется. Значение ключевых слов AR0-AR7 зависит от

значения выражения <выр> предшествующей директивы USING. Имена-синонимы для ключевых слов AR0-AR7 принимают их текущие значения, которые не изменяются последующими директивами USING.

Пример использования директивы USING:

```
USING 3
PUSH    AR2   ;сохранить содержимое регистра R2
              ;из банка рабочих регистров #3
. . .
USING 1
PUSH    AR2   ;сохранить содержимое регистра R2
              ;из банка рабочих регистров #1
```

12. Директива END

Форма записи директивы:

END

Данная директива определяет конец программного модуля. Предложения, следующие за директивой END, игнорируются.

III. ОБЪЕКТ ИССЛЕДОВАНИЯ, ПРИБОРЫ, ОБОРУДОВАНИЕ

Персональный компьютер.

IV. ЗАДАНИЕ НА РАБОТУ

Написать и отладить программы на ассемблере для МК51, реализующие следующие возможности:

1. Выполнение арифметических операций
- сложение (операнды однобайтовые, двухбайтовые);
- вычитание (операнды однобайтовые, двухбайтовые);
- умножение (операнды однобайтовые);
- деление (операнды однобайтовые);

2. Организация циклов.

3. Написание процедур
- процедура перемножения операндов (двухбайтовых);
- процедура деления операндов (однобайтовых);

V. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить систему команд.
2. Написать тексты программ.

VI. ОФОРМЛЕНИЕ ОТЧЕТА

Отчёт должен содержать:

- описание используемых операторов языка Ассемблер;
- алгоритмы и листинги программ.

VII. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие из микросхем не относятся к СБИС?

2. Что означает RISC – процессор?

3. В каких случаях современные операционные системы и системы программирования не используют переключение контекста процессора (содержимого регистров и отдельных управляющих триггеров)?

4. Какое основное архитектурное отличие мультискалярного процессора от суперскалярного?

5. На каком уровне системы управления решаются вопросы непосредственного управления исполнительными механизмами?

6. Какая архитектура микропроцессоров не являются RISC – процессорами?

7. Что показывает PR-рейтинг?

VIII. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики. - М.: Энергоатомиздат, 1987.- 304 с.
2. Кеннет Дж.Данхоф, Кэрол Л.Смит Основы микропроцессорных вычислительных систем.: Пер. с англ. А.А.Савельева. - М.: Высш. шк., 1986.- 288с.
3. Микропроцессоры. В 3-х кн. Кн.1.Архитектура и проектирование микроЭВМ. Организация вычислительных процессов.: Учеб. для втузов / П.В.Нестеров, В.Ф.Шаньгин, В.Л.Горбунов и др.; Под редакцией Л.Н.Преснухина. М.:Высш.шк.,1986.-495 с.
4. Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю.М.Казаринов, В.Н. Номоконов, Г.С. Подклетнов, Ф.В.Филиппов. Под ред. Ю.М.Казаринова.- М.: Высш. шк.,1990.-269с.
5. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления. Справочник.; Под общ. ред. С.Т.Хвоща.- Л.: Машиностроение. Ленингр. отд-ние, 1987.-639с.

Лабораторная работа № 2. Изучение функциональной схемы микроЭВМ на базе микроконтроллера МК 51.

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение функциональных и принципиальных схем микроЭВМ, выполненной на базе микроконтроллера МК51, и подключаемых к ней периферийных модулей.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1. МИКРО ЭВМ.

В данном разделе рассматривается пример построения МК-системы на основе ОМЭВМ i8051, которая может быть использована для приобретения навыков программирования и отладки программ для i8051, а также рассматриваться в качестве макета реальной системы управления для отладки ее прикладного программного обеспечения совместно с объектом управления в реальном масштабе времени.

Функциональная схема микроЭВМ представлена на рисунке 1.

Разработанная микроЭВМ содержит следующие средства:

-внешнюю память, реализованную на базе одной (или двух) микросхем статического ОЗУ, которая является общей памятью, как для хранения программ, так и данных.

-блок сопряжения с интерфейсной платой персонального компьютера, позволяет осуществлять операции чтения/записи из внешней памяти и работать с внешними устройствами, минуя процессор.

-шины адреса и данных, выводящиеся вместе с управляющими сигналами на внешние разъемы. Такая конструкция позволяет наращивать возможности системы путем подключения внешних плат расширения (контроллер клавиатуры-дисплея, АЦП, ЦАП и др.)

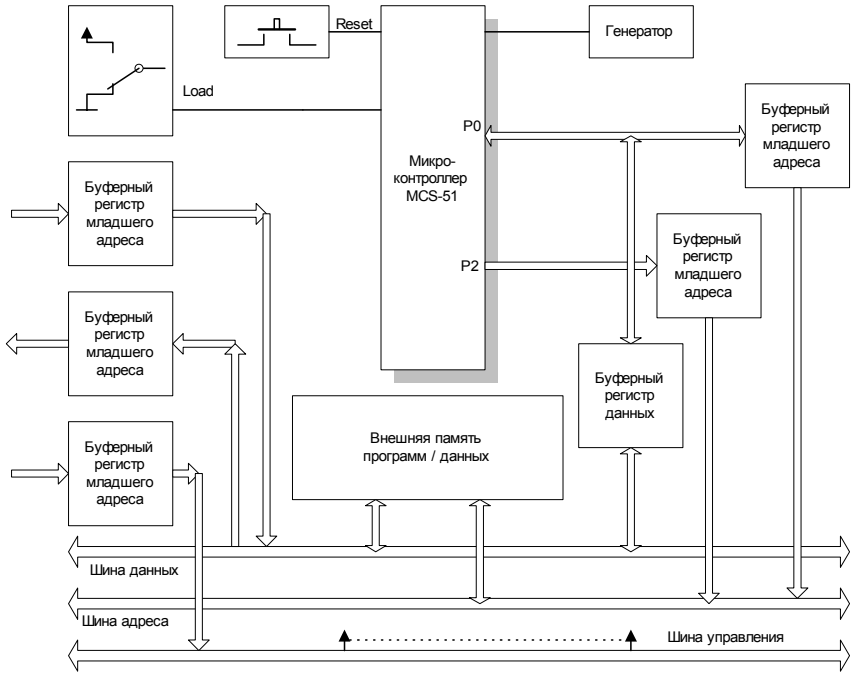


Рис. 1.а. Функциональная схема микроЭВМ (внешняя память данных реализована на одной микросхеме статического ОЗУ).

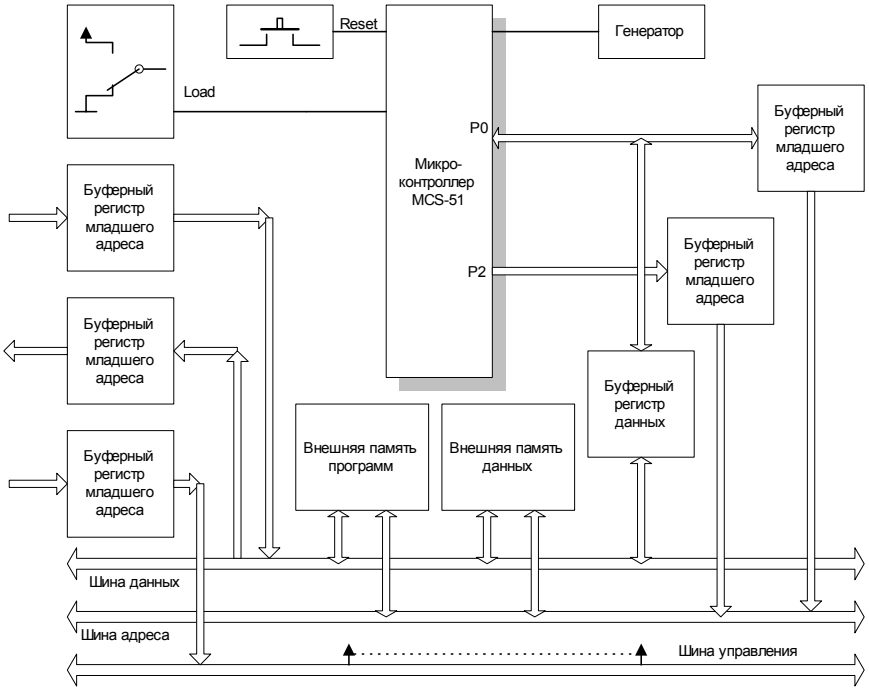


Рис. 1.б. Функциональная схема микроЭВМ (внешняя память данных реализована на двух микросхемах статического ОЗУ).

Функциональная схема возможного варианта вычислительного комплекса на базе разработанной платы микроЭВМ представлена на рисунке 2.

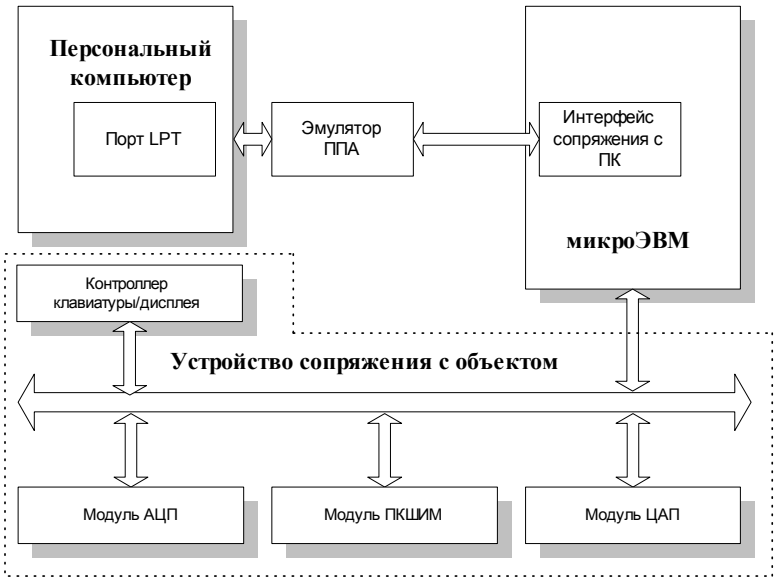


Рис. 2. Функциональная схема вычислительного комплекса на базе разработанной платы микроЭВМ.

Принципиальная электрическая схема микроЭВМ представлена в приложении.

МикроЭВМ имеет два режима работы: совместная работа с ПК и автономная работа, выбор нужного режима осуществляется переключателем Load.

1. Совместная работа с ПК необходима в любом случае для загрузки отлаживаемой программы во внешнюю память микроЭВМ. Также этот режим может применяться для считывания внешней памяти в ПК (например, если микроЭВМ занималась сбором данных) или непосредственной работы с внешними устройствами.
2. Режим автономной работы возможен после окончания загрузки программы в микроЭВМ и перевода переключателя 2 в соответствующее положение. При этом происходит принудительный сброс микроконтроллера, для очистки всех внутренних регистров и начала выполнения программы с нулевого адреса.

Блок сопряжения с PC выполнен на четырех регистрах КР580ИР82. Два из них выполняют функции буферов младшего и старшего байтов шины адреса, а два других являются буферами чтения и записи шины данных. При этом порт А интерфейсной платы работает на вывод, порт В – на ввод, порт С вырабатывает стробирующие сигналы. С порта С интерфейсной платы снимаются также сигналы EWR и ERD, дублирующие сигналы WR и RD микроконтроллера, благодаря чему возможна работа с внешними устройствами без участия микроконтроллера. Это может оказаться очень полезным при отладке внешних устройств, когда можно проверить их функционирование подавая на них управляющие сигналы и данные непосредственно с эмулятор ППА персонального компьютера.

Буферный регистр данных предназначен для удержания байта данных во время проведения операций чтения/записи во внешнюю память программ/данных. Физически он реализован на микросхеме К555АП6.

Буферные регистры младшего и старшего байтов адреса выполнены на микросхемах КР580ИР82. Первый из них стробируется сигналом ALE, второй работает в «сквозном» режиме, когда разрешении автономной работы микроЭВМ.

2.2. ЭМУЛЯТОР ППА.

Взаимодействие и обмен данными микроЭВМ с ПК осуществляется с помощью интерфейсного модуля выполненного с использованием параллельного периферийного адаптера (ППА) на микросхеме КР580ВВ55 (эмулятор ППА). Схема включения ППА представлена на рис. 3.

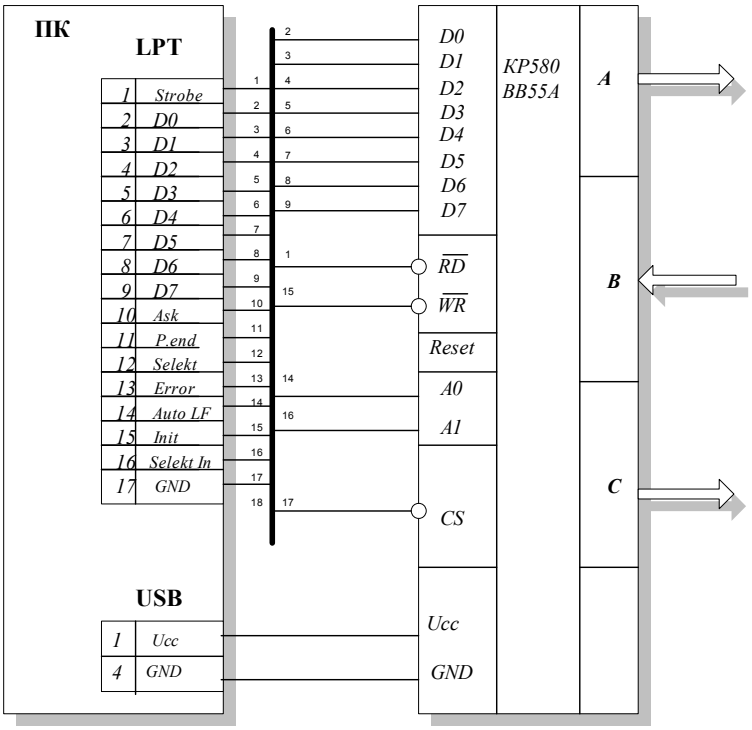


Рис. 3. Функциональная схема эмулятора ППА.

ППА подключается к LPT порту персонального компьютера. При этом необходимо иметь в виду, что схема ориентирована на использование расширенного параллельного порта, появившегося в компьютерах начиная с семейства PS/2.

В отличие от стандартного порта, расширенный LPT порт может работать как в режиме вывода байта, так и в режиме ввода байта. Протокол обмена формируется программно, а для указания направления передачи в регистр управления порта введен специальный бит CR.5: при CR.5 =0 буфер данных работает на вывод, CR.5 =1 – на ввод. Иногда этот порт называют enhanced bi-directional.

Фрагмент программы, управляющей портом LPT, представлен на рис. 4.

Рис. 4. Фрагмент программы, управления обменом информацией через порт LPT.

2.3. УСТРОЙСТВО СОПРЯЖЕНИЯ С ОБЪКТОМ.

Устройство сопряжения с объектом (УСО) предназначено для согласования уровня и формы сигналов, передаваемых от микроЭВМ к объекту управления (ОУ) и от ОУ к микроЭВМ.

Устройство сопряжения с объектом включает в свой состав три функционально независимых модуля: модуль преобразования цифрового кода в ШИМ- сигнал (модуль ПКШИМ), модуль преобразования аналогового сигнала в цифровой код (модуль АЦП), модуль преобразования цифрового кода в аналоговый сигнал (модуль ЦАП). Функционально в состав УСО может входить модуль управления жидко-кристаллическим индикатором (ЖКИ) и вводом сигналов с клавиатуры (контроллер клавиатуры/дисплея).

Принципиальная схема УСО представлена в приложении 2. Схема управления режимами работы модулей, входящих в УСО, реализована на микросхемах DD8, DD14, DD15 (Приложение 2).

2.3.1. МОДУЛЬ ПКШИМ.

Функциональная схема модуля ПКШИМ представлена на рис. 5.

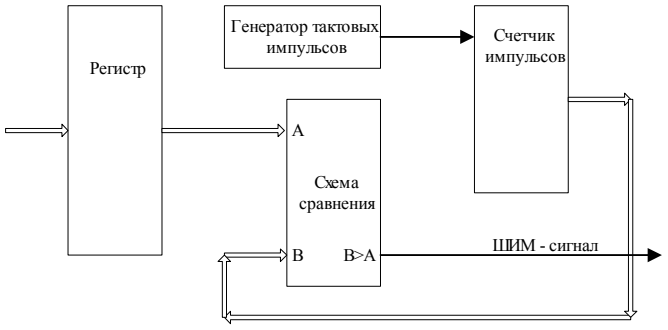


Рис.5. Функциональная схема модуля ПКШИМ.

Модуль реализован на микросхемах DD2, DD3, DD4, DD10 (Приложение 2).

2.3.2. МОДУЛЬ АЦП.

Функциональная схема модуля АЦП представлена на рис. 6.

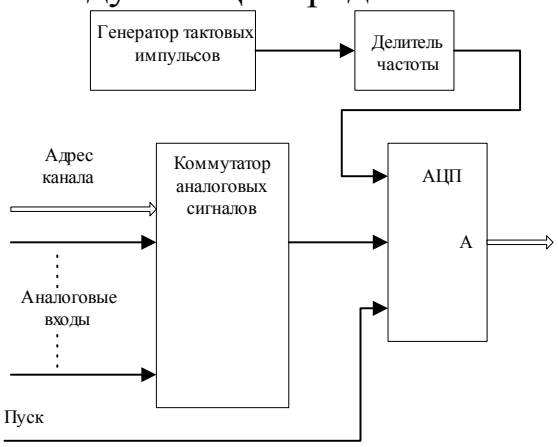


Рис.6. Функциональная схема модуля АЦП.

Модуль реализован на микросхемах DD7, DD10, DD12, DD13 (Приложение 2).

2.3.3. МОДУЛЬ ЦАП.

Функциональная схема модуля ЦАП представлена на рис. 7.

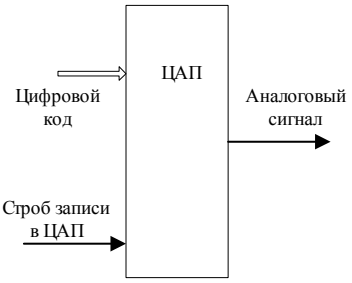


Рис.7. Функциональная схема модуля ЦАП.

Модуль реализован на микросхеме DD5 (Приложение 2).

III. ОБЪЕКТ ИССЛЕДОВАНИЯ, ПРИБОРЫ, ОБОРУДОВАНИЕ

В работе используется специализированный стенд, разработанный на базе микроЭВМ МК51, осциллограф, персональный компьютер. Для сопряжения микроЭВМ и персонального компьютера используется специальный шлейф, подключаемый к стандартному параллельному порту персонального компьютера.

Общий вид стенда представлен на рис. 8. Общий вид печатной платы микроЭВМ представлен на рис.9. Общий вид эмулятора параллельного периферийного адаптера представлен на рис.10.

Рис. 8 Общий вид микроЭВМ.

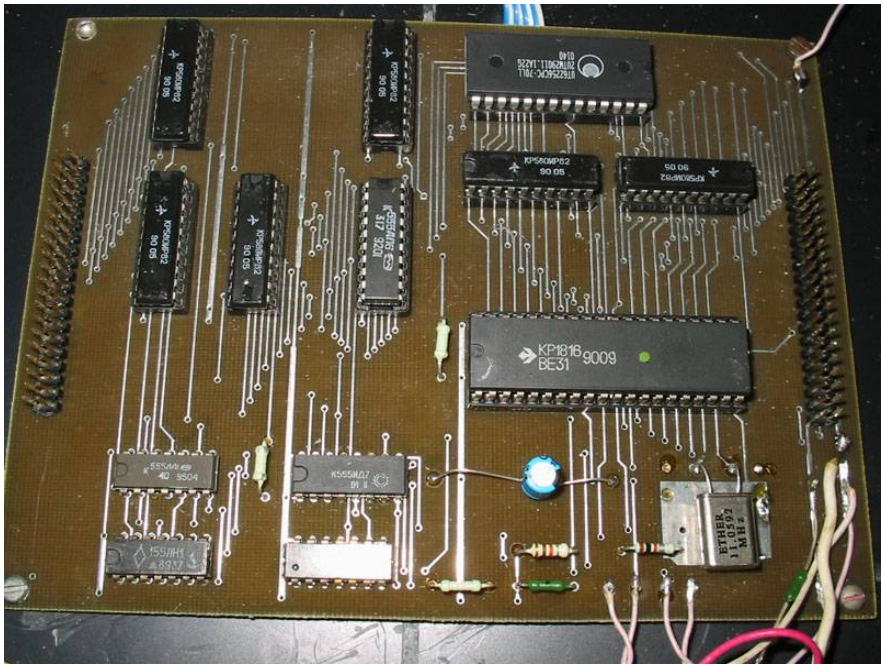


Рис. 9 Печатная плата микроЭВМ.

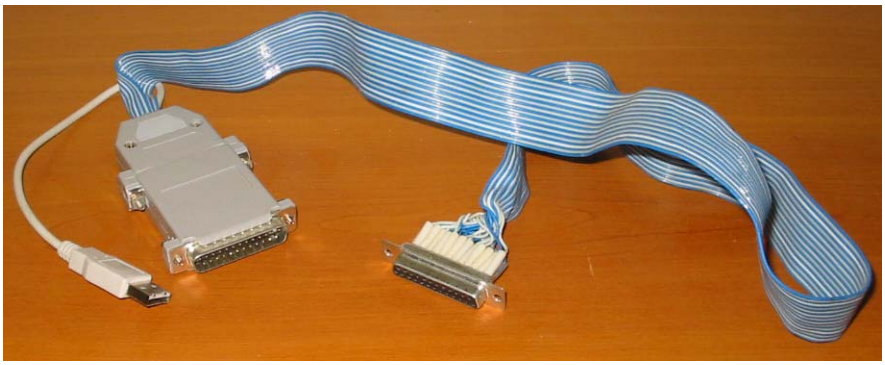


Рис. 10. Эмулятор параллельного периферийного адаптера на базе стандартного параллельного порта.

IV. ЗАДАНИЕ НА РАБОТУ

Изучить функциональную схему микроконтроллера.

Изучить принципиальную схему микроконтроллера и функциональные возможности используемых микросхем.

Изучить протоколы обмена информацией микроконтроллера с памятью и персональным компьютером.

Построить диаграммы сигналов управления, сопровождающих обмен информацией.

Изучить модуль сопряжения стандартного параллельного порта персонального компьютера с микроЭВМ.

V. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить функциональную схему микроконтроллера.
2. По принципиальной схеме контроллера определить наименования портов, используемых для обмена информацией.
3. Изучить названия линий управления, используемых для управления обменом информацией внутри микроконтроллера.
4. Изучить способ формирования сигналов управления для обмена информацией микроконтроллера с персональным компьютером и внешними устройствами.
5. Изучить режимы работы микроЭВМ.
6. Составить диаграммы сигналов, соответствующих режимам обмена информацией с модулями УСО и микроЭВМ.
7. По указанию преподавателя составить функциональную схему модуля ввода сигналов с клавиатуры 4*4 или вывода сигналов на ЖКИ.
8. Составить алгоритм обмена информацией персонального компьютера и микроЭВМ при помощи рассмотренного эмулятора.

VI. ОФОРМЛЕНИЕ ОТЧЕТА

В отчете должны быть представлены функциональная схема микроЭВМ, диаграммы управляющих сигналов, описание протоколов обмена информацией микроЭВМ с персональным компьютером и с внешними устройствами, функциональная схема эмулятора параллельного периферийного адаптера на базе стандартного параллельного порта.

VII. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Охарактеризуйте систему команд микроконтроллера МК51.
2. Какой тип архитектуры имеет микроконтроллер МК51?
3. Сколько микросхем памяти необходимо для хранения кодов команд и данных в предложенной функциональной схеме?

4. Какой тип памяти используется в микроЭВМ и почему?
5. Каким образом производится разделение пространства памяти на память данных и память программ?

VIII. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Горячев О.В. Микропроцессоры и микропроцессорные вычислительные системы. – Тула, ТулГУ, 1998.
2. Микропроцессорные автоматические системы регулирования. Основы теории и элементы./Под ред. В.В.Солодовникова,- М., 1991.
3. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат. 1990г. - 224с.

Лабораторная работа № 3. Написание и отладка программ на языке ассемблер. Оболочка КИТ.

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение технологии написания и отладки программ для микроЭВМ, выполненной на базе микроконтроллера МК51

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Язык ассемблера АСМ51 предназначен для использования в системах программирования однокристальных микро-ЭВМ серии КР1816, совместимых по структуре и системе команд с однокристальной микро-ЭВМ КР1816ВЕ51 (ОЭВМ серии КР1816).

Средства языка ассемблера АСМ51 обеспечивают модульный способ построения программы.

Программный модуль, записанный на языке ассемблера, называется исходным модулем. Результат трансляции исходного модуля - объектный модуль.

Исходная программа представляет собой последовательность предложений языка, сгруппированных в сегменты и оформленных в виде файла. Сегменты исходного модуля содержат описание машинных команд, данных, адресов и служат для генерации абсолютных или перемещаемых сегментов объектного модуля.

В зависимости от устройства памяти и способа адресации данных в этой памяти сегменты исходного модуля делятся на 4 типа:

CODE - используется для определения команд, данных и адресов в пространстве памяти команд;

XDATA - используется для определения адресов в пространстве внешней памяти данных;

DATA - используется для определения прямых адресов в пространстве внутренней памяти данных;

IDATA - используется для определения косвенных адресов в пространстве внутренней памяти данных;

BIT - используется для определения прямых побитовых адресов в пространстве внутренней памяти данных.

При описании перемещаемого сегмента ему присваивается имя и тип, а также определяется способ его объединения с одноименными сегментами, описанными в других исходных модулях. Абсолютным сегментам присваивается только тип. Они не могут быть объединены с другими сегментами.

В языке ассемблера имеются средства для организации символических ссылок между различными программными модулями (т.е. Средства определения общих и внешних символических имен).

ОПИСАНИЕ СТАНДАРТНОГО ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ.

Программа в общем случае строится из набора абсолютных и (или) перемещаемых объектных модулей и представляет собой абсолютный объектный модуль.

Ассемблер x8051.exe

Осуществляет преобразование исходного модуля, представляющего собой программу на языке ассемблера ASM51, в объектный код.

Пример:

>x8051.exe test.asm

В этом случае произойдет компиляция модуля test.asm, и в случае отсутствия ошибок, будет создан файл, содержащий объектный код test.obj.

Линкер link.exe

Осуществляет преобразование одного или нескольких объектных модулей в программу, готовую к загрузке в ПЗУ микроконтроллера. Всю информацию о необходимых действиях линкер получает из командного файла, который предятавляет из себя последовательность имен объекных модулей с указанием смещения каждого из них в адресном пространстве.

Например, если имеется файл test.cmd, содержащий следующие строки:

Test

0100

то выполнение команды

>link.exe test.cmd

приведет к созданию файла test.hex, содержащий код программы объектного модуля test.obj начиная с адреса 0100h.

ОПИСАНИЕ СПЕЦИАЛЬНОГО ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ.

Программа Tasmed51.

Предназначена для чтения/записи внешней памяти программ (ВПП) платы МК с использованием интерфейсной платы на базе микросхемы К580ВВ55. На данный момент реализованы следующие функции:

1. Программирование LPT – порта и инициализация параллельного периферийного адаптера КР580ВВ55А.
2. Чтение с диска программ для МК51 записанных в следующих форматах:
   - bin – бинарный файл размером 2048 байт (2Kb)
   - bintxt – текстовое представление бинарного файла
3. Загрузка данных (программ) во ВПП микроЭВМ.
4. Чтение ВПП микроЭВМ и отображение ее на экране.

Программа инициализации LPT – порта и параллельного периферийного адаптера КР580ВВ55А представлена на рис. 1.

```
Uses crt,dos;

var
  h, m, hund,ne : Word;
  i,j:Word;
  n:LongInt;
  f:File Of Byte;
  s,s1,s2,s3:String;
  b,b1:Byte;
  A:array[0..33000] of byte;
  nrow,nb,StartAdress,code:word;

function Dec2Hex(D:integer):string;
Const s16:Array[0..15] of char =
('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');
Var
      ss:string;
begin

      ss:='';
      repeat
            ss:=s16[d mod 16]+ss;
      d:=d div 16;
   until d=0;
   if length(ss)=1 then ss:='000'+ss;
   if length(ss)=2 then ss:='00'+ss;
   if length(ss)=3 then ss:='0'+ss;
   Dec2Hex:=ss+'H';
end;

procedure LoadHexfile(HexFile:String);
Var
  F:text;
  S,s1: string;
```

```
  B: Byte;
begin
   For i:=0 to 33000 Do Begin
      A[i]:=0;
   End;
   n:=0;
   Assign(F,HexFile);
   Reset(F);
   while not(eof(F)) do begin
               Readln(F, S);
      If s<>':00000001FF' then Begin
         Delete(S,1,1);
{          nb:=StrToInt('$'+Copy(S,1,2));}
         Val('$'+Copy(S,1,2),nb,code);
{          StartAdress:=StrToInt('$'+Copy(S,3,4));}
         Val('$'+Copy(S,3,4),StartAdress,code);
         For j:=1 to nb Do Begin
{             A[StartAdress+j-1]:=StrToInt('$'+Copy(S,9+(j-1)*2,2));}
            Val('$'+Copy(S,9+(j-1)*2,2),A[StartAdress+j-1],code);
         End;
         n:=StartAdress+nb;
{          Application.ProcessMessages;}
      end;
   end;

{   DisplayMemArray(n);}

end;

{++++++++++++ Access to I/O ports ++++++++++}

Procedure Delay2(i:word);
var l,j,k:longint;
begin
      l:=10*i;
      for j:=1 to l do begin
      k:=round(sin(1));
   end;
end;

Procedure init;
begin
      asm
            mov dx,303h
            mov al,82h
```

```
                out dx,al
        end;
end;

Procedure set_a(b:byte);
begin
  asm
        mov dx,300h
        mov al,b
        out dx,al
  end;
end;

Procedure set_c(b:byte);
begin
  asm
        mov dx,302h
        mov al,b
        out dx,al
  end;
end;

Function get_b:byte;
var b:byte;
begin
  asm
        mov dx,301h
        in  al,dx
        mov b,al
  end;
  get_b:=b;
end;

Procedure init_mb;
begin
{       set_a($00);
        delay(1);
        set_c($70);
        delay(1);}
        set_a($00);
        delay(1);
        set_c($30);
        delay(1);
end;
```

```
Procedure set_bus(b:byte);
begin
{       set_a(b);
        delay(1);
        set_c($40);
        delay(1);
        set_c($41);
        delay(1);
        set_c($40);
        delay(1);}

        set_a(b);
        delay(1);
        set_c($0C);
        delay(1);
        set_c($08);
        delay(1);

end;

Procedure set_loaddr(b:byte);
begin
{       set_a(b);
        delay(1);
        set_c($72);
        delay(1);
        set_c($70);
        delay(1);}

        set_a(b);
        delay(1);
        set_c($09);
        delay(1);
        set_c($08);
        delay(1);

end;

Procedure set_hiaddr(b:byte);
begin
{       set_a(b);
        delay(1);
        set_c($74);
        delay(1);
        set_c($70);
```

```
        delay(1);}

        set_a(b);
        delay(1);
        set_c($0A);
        delay(1);
        set_c($08);
        delay(1);

end;

Function get_bus:byte;
var b:byte;
begin
{       set_c($38);
        delay(1);
        set_c($30);
        delay(1);
        b:=get_b;
        delay(1);
        get_bus:=b;}

        set_c($30);
        delay(1);
        set_c($38);
        delay(1);
        b:=get_b;
        delay(1);
        get_bus:=b;
end;

Procedure write_pm(addr:word;b:byte);
var loaddr,hiaddr:byte;
begin
        asm
                mov ax,addr
                mov   loaddr,al
                mov   hiaddr,ah
        end;
        set_loaddr(loaddr);
        set_hiaddr(hiaddr);

{       set_c($60);
        delay(1);
```

```
        set_bus(b);

        set_c($00);
        delay(1);
        set_c($40);
        delay(1);
        set_c($60);
        delay(1);
        set_c($70);
        delay(1);}

        set_a(b);
        delay(1);
        set_c($0C);
        delay(1);
        set_c($2C);
        delay(1);
        set_c($0C);
        delay(1);
        set_c($08);
        delay(1);

end;

Function read_pm(addr:word):byte;
var loaddr,hiaddr,b:byte;
begin
        asm
                mov   ax,addr
                mov   loaddr,al
                mov   hiaddr,ah
        end;
        set_loaddr(loaddr);
        set_hiaddr(hiaddr);

{       set_c($30);
        delay(1);

        b:=get_bus;

        set_c($70);
        delay(1);

        read_pm:=b;}
```

```
        set_c($18);
        delay(1);
        set_c($10);
        delay(1);
        b:=get_b;
        delay(1);
        set_c($18);
        delay(1);
        set_c($08);
        delay(1);

        read_pm:=b;

end;

Begin

 ne:=0;

 If ParamStr(1)<>'' Then Begin

    Init;
    Init_mb;

    s:=ParamStr(1);

    LoadHexFile(S);
{    n:=nb;}

{    Assign(f,s);
    Reset(f);
    n:=FileSize(f);}

    WriteLn;

    write('Loading file ');
    TextColor(15);
    Write(s);
    TextColor(7);
    Write(' Size=');
    TextColor(15);
    Write(n);
    TextColor(7);
    WriteLn(' bytes');
```

```
   WriteLn;

   For i:=0 to n-1 Do Begin

{       Read(f,b);}
       b:=a[i];

       Gotoxy(1,25);
       TextColor(7);
       Write(' loading byte ');
       TextColor(15);
       Write(i);
       TextColor(7);

       s1:='';
       For j:=0 to 49 do begin
          If j/50<i/n Then s1:=s1+chr($b2) Else s1:=s1+chr($b0);
       End;
       s1:='['+s1+']';

       Write(' ',s1);
{       TextColor(15);
       Write((i/n*100):2:2,'%');}

{       WriteLn(i,'  ',b);}

{       Delay(500);}

       Write_pm(i,b);

       b1:=Read_pm(i);

{       WriteLn(i,'  ',b1);}

       While (b1<>b) Do Begin
             ne:=ne+1;
             Gotoxy(1,25);
             TextColor(7);
             Write('Error ');
             TextColor(15);
             Write(ne);
             TextColor(7);
             Write(' write in adress: ');
             TextColor(15);
             WriteLn(Dec2Hex(i),'                                    ');
```

```
            Write_pm(i,b);
            b1:=Read_pm(i);
        End;

    End;

{   For i:=0 to 10 Do Begin

      Read(f,b);

      b:=Read_pm(i);
      WriteLn(i,b);
    End;}

{   Close(f);}

    writeln;
    TextColor(7);
    write('Loading successfull, ');
    TextColor(15);
    Write(ne);
    TextColor(7);
    Write(' errors, ');
    TextColor(15);
    Write((ne/n*100):2:2,'%');

  End;

end.
```

Рис. 1. Листинг программы инициализации LPT и ППА.

ПРИМЕРЫ ПРОСТЫХ ПРОГРАММ ФОРМИРОВАНИЯ СИГНАЛОВ ДЛЯ МИКРОЭВМ.

1. Формирование линейно нарастающего сигнала.

Код линейно нарастающего сигнала формируется на выводах порта 1.

Текст программы приведен на рис.2.

```
org 0000h          ;абсолютное смещение программы в адресном ;пространстве
                   микроконтроллера.
    MOV 90H,#00H   ;обнуление порта 1.
                   ;90H – адрес порта 1.
```

```
MainLoop:
        INC   90H          ; увеличение значения  порта 1 на единицу
        CALL  DELAY        ; вызов подпрограммы формирования
                           ;временной  задержки.
        JMP   MainLoop     ;зацикливание программы

DELAY:                     ;подпрограмма формирования временной  задержки
        MOV   R2,#255      ; запись константы в рабочий регистр
C1:
        DJNZ  R2,C1        ; программный цикл выполняющий задержку
        RET                ; возврат из подпрограммы
```

Рис.2.

Величину временной задержки в указанной программе не трудно вычислить исходя из времени выполнения команд микроЭВМ, т.к. каждая из команд **MOV, DJNZ** и **RET** выполняется за 2 машинных цикла, то общее время составит 2+2+2*R2+2 машинных цикла. Здесь учтено, что команда **DJNZ** выполняется в цикле R2 раз и то, что данная программа оформлена как процедура на вызов которой (по команде CALL) потребуется еще 2 машинных цикла. Таким образом общая задержка в приведенном примере составит 2+2+2*255+2=506 машинных циклов. Для тактовой частоты 12 МГц время выполнения одного машинного цикла составляет около 1 мкс, т.е время задержки составит 500 мкс. Это значение также является максимальной величиной задержки, которую можно получить с помощью данной подпрограммы.

2. Формирование синусоидального сигнала.

Код линейно нарастающего сигнала формируется на выводах порта 1.

Текст программы приведен на рис.3.

```
ORG 0000H
        MOV     DPTR, #SinCode    ; загрузка в регистр указатель
                                  ; смещение таблицы значений синуса
        MOV     R1,#0             ; обнуление рабочего регистра,
                                   ; который в данном случае является
                                   ; индексом выбираемого из таблицы
                                   ; элемента.
START:
        MOV     A,R1
        MOVC    A,@A+DPTR         ; запись в аккумулятор значения
                                   ; из таблицы
        MOV     90H,A             ; пересылка аккумулятора в порт А
        MOV     R2,#0FFH
```

```
C1:   DJNZ    R2,C1          ; организация временной задержки
      INC     R1             ; инкрементирование индексного
                             ; регистра
      JMP   START            ; зацикливание программы

ORG 0600H                    ; смещение таблицы значений синуса в
                             ; адресном пространстве МК

SinCode:                     ; таблица, содержащая 256 значений синуса
      DB 158, 161, 164, 167, 170, 173, 176, 179, 181, 184
      DB 187, 190, 193, 195, 198, 200, 203, 205, 208, 210
      DB 213, 215, 217, 219, 221, 223, 225, 227, 229, 231
      DB 233, 235, 236, 238, 239, 241, 242, 243, 245, 246
      DB 247, 248, 249, 250, 250, 251, 252, 252, 253, 253
      DB 253, 254, 254, 254, 254, 254, 254, 254, 253, 253
      DB 252, 252, 251, 251, 250, 249, 248, 247, 246, 245
      DB 244, 243, 241, 240, 239, 237, 235, 234, 232, 230
      DB 228, 226, 224, 222, 220, 218, 216, 214, 211, 209
      DB 207, 204, 202, 199, 196, 194, 191, 188, 186, 183
      DB 180, 177, 174, 171, 168, 166, 163, 159, 156, 153
      DB 150, 147, 144, 141, 138, 135, 132, 129, 125, 122
      DB 119, 116, 113, 110, 107, 104, 101, 98, 95, 91
      DB 89, 86, 83, 80, 77, 74, 71, 68, 66, 63
      DB 60, 58, 55, 52, 50, 47, 45, 43, 40, 38
      DB 36, 34, 32, 30, 28, 26, 24, 22, 20, 19
      DB 17, 15, 14, 13, 11, 10, 9, 8, 7, 6
      DB 5, 4, 3, 3, 2, 2, 1, 1, 0, 0
      DB 0, 0, 0, 0, 0, 1, 1, 1, 2, 2
      DB 3, 4, 4, 5, 6, 7, 8, 9, 11, 12
      DB 13, 15, 16, 18, 19, 21, 23, 25, 27, 29
      DB 31, 33, 35, 37, 39, 41, 44, 46, 49, 51
      DB 54, 56, 59, 61, 64, 67, 70, 72, 75, 78
      DB 81, 84, 87, 90, 93, 96, 99, 102, 105, 108
      DB 111, 115, 118, 121, 124, 127
```

В данном примере синусоидальный сигнал формируется в процессе выборки отмасштабированных и смещенных значений синуса из таблицы, располагающейся в памяти программ микроконтроллера. –1 соответствует значение 0, а +1 – 255. Всего таблица содержит 256 элементов соответствующих значениям синуса в диапазоне от 0 до 360 градусов.

Если есть необходимость сократить объем памяти, отводящейся под хранение таблицы значений синуса, сохранив при этом тот же шаг, то этого можно достичь, оставив первые 64 байта таблицы, соответствующие диапазону от 0 до 90 градусов, а все остальные значения получать, путем небольшого

усложнения программы, из этой таблицы. Однако, следует иметь в виду, что это повлечет за собой снижение быстродействия в 1.5...2 раза.

III. ОБЪЕКТ ИССЛЕДОВАНИЯ, ПРИБОРЫ, ОБОРУДОВАНИЕ

В работе используется специализированный стенд, разработанный на базе микроЭВМ МК51, осциллограф, персональный компьютер. Для сопряжения микроЭВМ и персонального компьютера используется специальный шлейф, подключаемый к стандартному параллельному порту персонального компьютера.

IV. ЗАДАНИЕ НА РАБОТУ

Изучить технологию отладки программ и записи отлаженных программ в память микроЭВМ, выполненной на базе однокристального микроконтроллера МК51.

С помощью встроенного текстового редактора ввести в память персонального компьютера программы, приведенные в тексте данной лабораторной работы. Отладить программы. Записать исполняемый код в память микроЭВМ. Выполнить программы.

Написать процедуру вывода с помощью модуля АЦП аналогового сигнала, код которого формируется микроЭВМ. Подключить осциллограф и снять диаграмму изменения сигнала на выходе АЦП.

V. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

9. Изучить стандартное программное обеспечение, используемое для ввода и отладки программ для микроЭВМ.
10. Изучить специализированное программное обеспечение, используемое для передачи исполняемого кода в память микроЭВМ.
11. Познакомиться с программами, листинги которых представлены в данной работе. Составить алгоритм выполнения программ.
12. Ввести с помощью встроенного редактора программы представленные в тексте лабораторной работы в память персонального компьютера.
13. Выполнить трансляцию и ассемблирование программ. При обнаружении ошибок на любой из ступеней обработки, ввести необходимые исправления в исходные тексты программ.
14. Записать исполняемые коды программ в память микроЭВМ.
15. Выполнить программы.
16. Составить алгоритм вывода синусоидального сигнала, код которого формируется микроЭВМ, с помощью модуля АЦП.
17. Написать программу формирования аналогового синусоидального сигнала с помощью микроЭВМ.

18. Ввести с помощью встроенного редактора программу в память персонального компьютера.
19. Выполнить трансляцию и ассемблирование программы. При обнаружении ошибок на любой из ступеней обработки, ввести необходимые исправления в исходные тексты программ.
20. Записать исполняемые коды программ в память микроЭВМ.
21. Выполнить программу.
22. Подключить осциллограф к соответствующим выводам на корпусе микроЭВМ с целью наблюдения сигнала.
23. Составить отчет о проделанной работе.

VI. ОФОРМЛЕНИЕ ОТЧЕТА

В отчете должны быть представлены алгоритмы программ, листинги программ. Функциональная схема подключения осциллографа к микроЭВМ, диаграмма сигнала, полученная с помощью осциллографа.

VII. КОНТРОЛЬНЫЕ ВОПРОСЫ

6. Перечислите основные этапы отладки программного обеспечения с помощью разработанного лабораторного комплекса.
7. Каким образом можно ввести текст исходной программы в память персонального компьютера?
8. Как и зачем выполняется ассемблирование программы?
9. Что такое редактирование программы и каким образом оно выполняется?
10. Каким образом записывается программа в память микроЭВМ?
11. Как запускается программа на исполнение?
12. Каким образом формируются циклы в микроЭВМ, приведите примеры?
13. Дайте описание команды:

    MOVC A,@A+DPTR

    Ее назначение и выполняемые действия?
14. Перечислите основные сигналы, которые необходимо сформировать для управления модулем АЦП?
15. Основные характеристики модуля АЦП?

VIII. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

4. Горячев О.В. Микропроцессоры и микропроцессорные вычислительные системы. – Тула, ТулГУ, 1998.
5. Микропроцессорные автоматические системы регулирования. Основы теории и элементы./Под ред. В.В.Солодовникова,- М., 1991.
6. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат. 1990г. - 224с.

Лабораторная работа № 4. Написание драйверов для микроконтроллера и их отладка (на примере драйвера для фотоимпульсного датчика).

I. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение организации интерфейса микроЭВМ для управления электроприводами постоянного тока. Изучение фотоимпульсного датчика угла поворота. Написание драйвера для формирования цифрового кода по сигналам фотоимпульсного датчика.

II. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Функциональные схемы цифровых электрических следящих приводов постоянного тока (ЦЭСП) весьма разнообразны. Однако любой ЦЭСП может быть представлен в виде совокупности аналоговой и цифровой частей. Аналоговая часть, как правило, объединяет силовую систему привода – усилитель мощности, исполнительный двигатель, механическую передачу, аналоговые датчики. Цифровая часть в настоящее время наиболее часто представлена однокристальным микроконтроллером или одноплатной микроЭВМ, выполненной на базе однокристального контроллера. МикроЭВМ может выполнять по отношению к следящей системе функции задающего, сравнивающего и корректирующего устройств. По характеру связи между следящей системой и микроЭВМ принято разделять ЦЭСП на автономные и неавтономные.

Под автономной понимают следящую систему, в которой ЭВМ служит лишь в качестве источника входной информации, т. е. выполняет функции задающего устройства. Вычисление сигнала ошибки, цифровая коррекция динамических свойств осуществляется при этом дополнительным автономным вычислительным устройством.

Если сравнение задаваемого кода и кода сигнала обратной связи происходит в самой ЭВМ, то такая система называется неавтономной. При этом задача цифровой коррекции также возлагается на микроЭВМ.

Функциональная схема ЦЭСП с микроЭВМ в контуре управления представлена на рис.1.

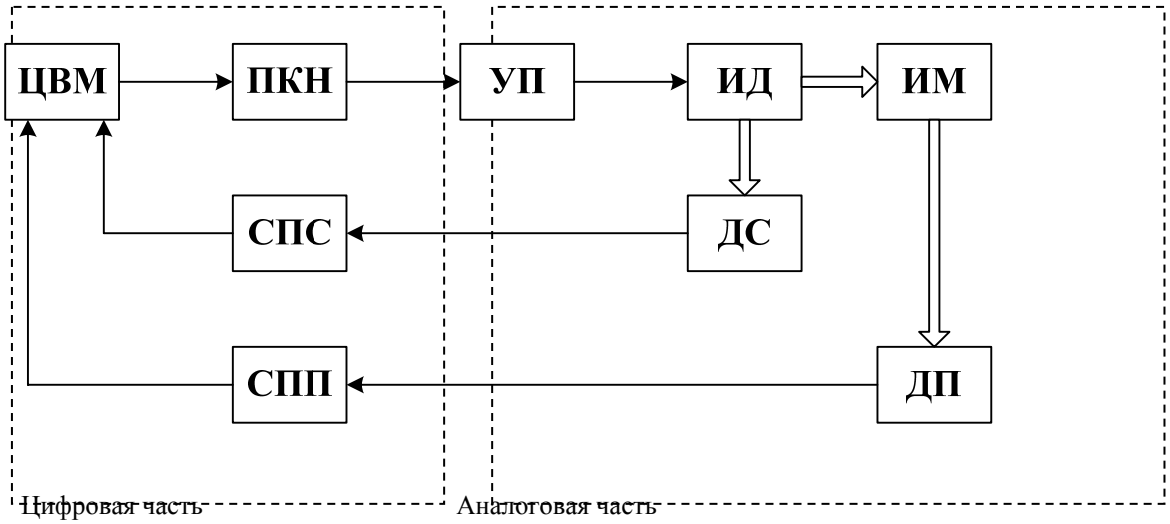


Рис.1.Функциональная схема ЦЭСП

Сигнал рассогласования поступает на преобразователь кода в напряжение ПКН, проходит усилитель-преобразователь УП (усилитель мощности) и подается на исполнительный двигатель ИД, который перемещает исполнительный механизм ИМ. С валом двигателя как правило связан датчик скорости ДС. На выходном валу исполнительного механизма ИМ устанавливают датчик положения ДП. С помощью схем преобразования сигналов скорости и положения СПС и СПП с датчика скорости ДС и датчика положения ДП сигналы преобразуются в двоичные коды, пропорциональные скорости вращения выходного вала привода и угловому перемещению исполнительного механизма. Датчик скорости со схемой преобразования представляет собой по существу преобразователь скорость-код (ПКС), а датчик положения – преобразователь положение-код (ПКП).

В современных приводах, как правило, в качестве датчиков положения и скорости используют различные фотоимпульсные датчики (ФИД), обладающие высокой стабильностью характеристик, отсутствием шумов, высокой точностью. Кроме того, импульсы, формируемые ФИД, весьма удобны для обработки микроЭВМ, что позволяет существенно упростить аппаратную часть системы управления.

В качестве ПКН используют различные схемы формирования широтно-импульсного сигнала. При этом, с ростом мощности микроконтроллеров, функции преобразования цифрового кода в ШИМ сигнал также выполняет микроконтроллер.

НАПИСАНИЕ ДРАЙВЕРА ДЛЯ ФОТОИМПУЛЬСНОГО ДАТЧИКА.

ФИД предназначен для преобразования угла поворота вала двигателя в две последовательности прямоугольных импульсов S_1, S_2 количество которых

пропорционально углу поворота, а частота – частоте вращения вала двигателя. Кроме того, на каждый оборот диска ФИД формируется один импульс S_3 – называемый импульсом сброса. Указанный импульс используется для обнуления счетчиков, применяемых при формировании цифрового кода угла по двум указанным выше последовательностям импульсов.

Принцип действия фотоимпульсного датчика основан на модуляции светового потока вращающимся металлическим диском. Диск синхронизатора (см. рис.) имеет две информационные дорожки и одну дорожку для формирования импульса сброса. Количество отверстий на информационных дорожках варьируется от 64 и до нескольких тысяч. Дорожка для формирования импульса сброса имеет одно отверстие.

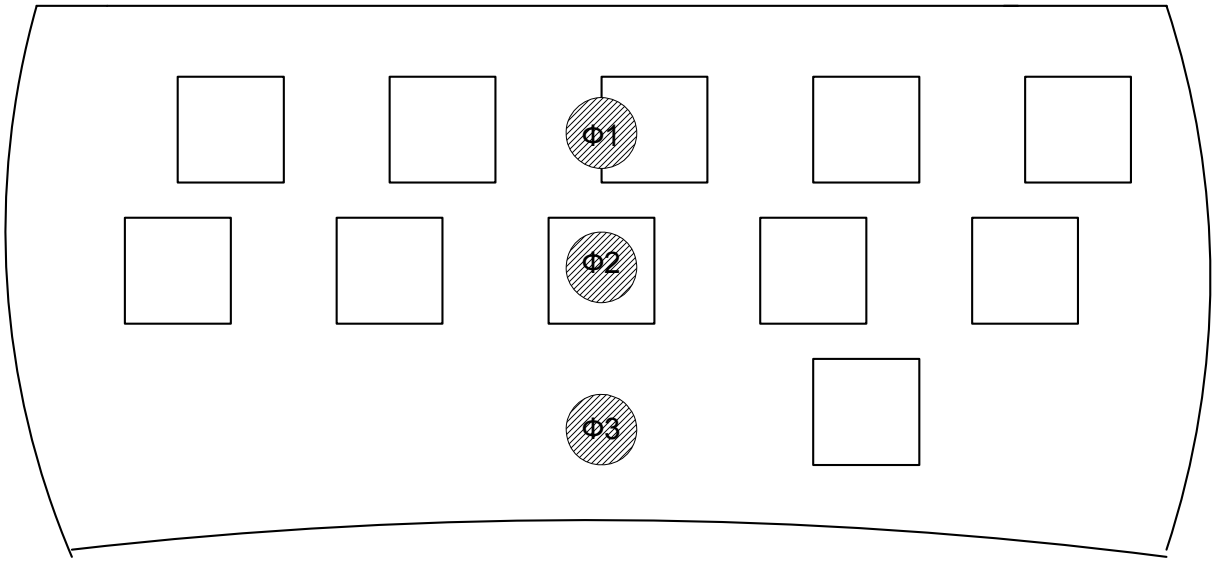


Рис. Элемент кодирующего диска ФИД с фотоимпульсными датчиками Ф1, Ф2 и Ф3.

Каждый фотоимпульсный датчик включает в себя излучатель и фотодиод, расположенных напротив друг друга, с различных сторон диск.

Очевидно, что при подобном расположении фотоприемников сигналы S_1 и S_2 сдвинуты по фазе относительно друг друга на четверть периода. Количество импульсов указанных сигналов определяется количеством окон на кодовом диске. Например, для ФИД _____, имеет для формирования каждой последовательности 1024 окошка. Сигнал S_3 возникает однажды при полном повороте кодирующего диска и осуществляет коррекцию начала отсчета.

Сигналы ФИД в виде последовательности импульсов показаны на рис. Диаграмма на рис.__а соответствует вращению диска по часовой стрелке; диаграмма на рис. ___ б - вращению диска против часовой стрелки

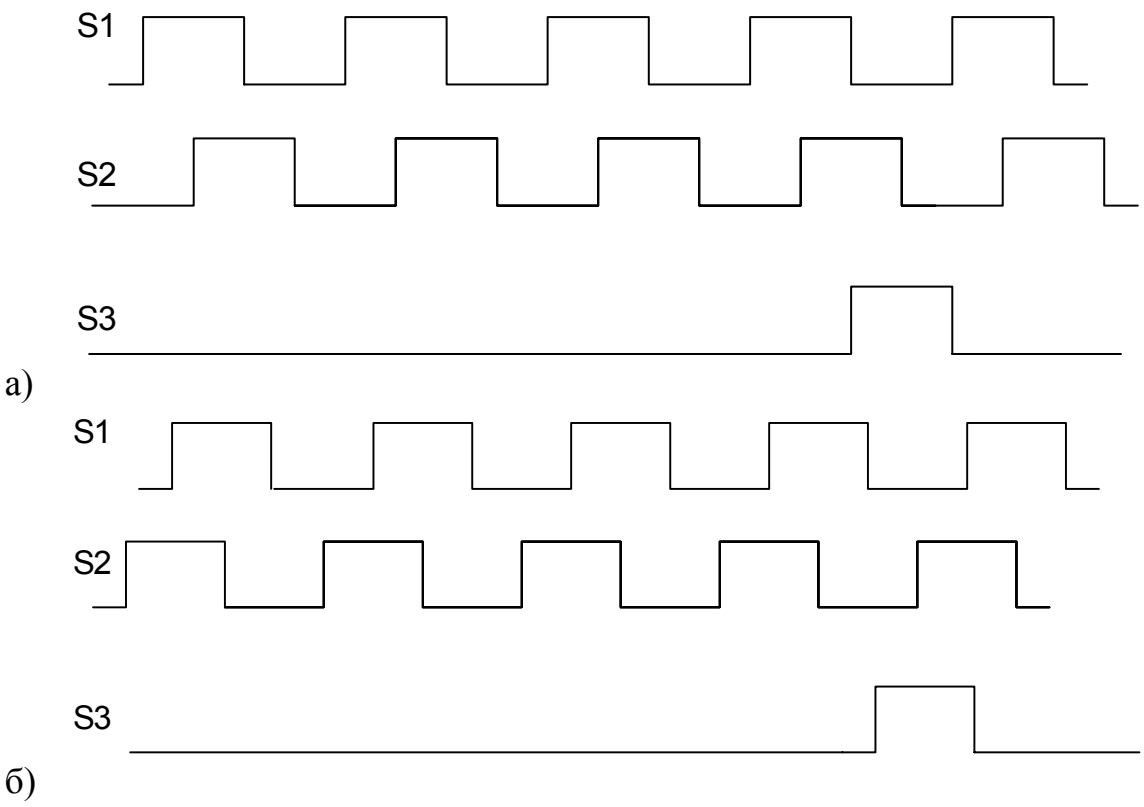


Рис. Диаграммы сигналов ФИД.

Таким образом, как видно из приведенных диаграмм, фаза сигнала S2 по отношению к S1 зависит от направления вращения (+/- четверть периода), следовательно, рассматриваемый датчик может быть использован для построения реверсивных систем. Информация о направлении вращения кодового диска должна формироваться на основе анализа фазового сдвига сигналов S_1 и S_2 друг относительно друга.

Если по изменению уровня сигналов S_1 и S_2 производить инкремент или декремент (в зависимости от фазового сдвига между S_1 и S_2) счетчика угла, то полному обороту диска ФИД будут соответствовать 1024 импульса, что соответствует точности представления углового положения $360^0/1024 = 0{,}351^0$. Точность датчика может быть повышена путем учетверения счетных импульсов. Для нашего примера точность составит $0{,}0878^0$ или 1,5 мрад.

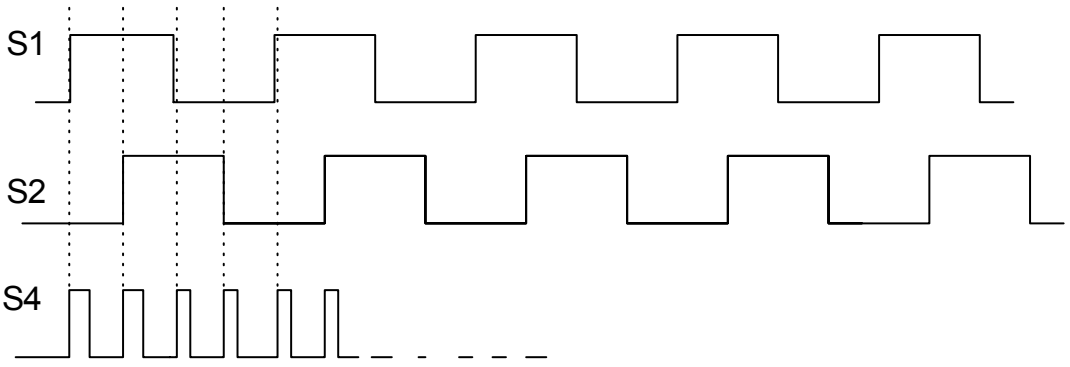


Рис. Диаграммы сигналов ФИД после учетверения счетных импульсов.

Сигнал S_3 используется для коррекции значения счетчика угла. Если коррекция производится только фронтом сигнала S_3, при смене направления вращения возникает ошибка, обусловленная сменой начала отсчета из-за ненулевой длительности импульса S_3. Чтобы исключить возникновение ошибки, в момент коррекции необходимо определять направление вращения диска синхронизатора и при необходимости изменять наступление события коррекции с фронта на срез сигнала S_3.

Частота вращения вала двигателя определяется по величине угла поворота за интервал времени dt. Полному обороту диска синхронизатора соответствует 4096 импульсов. Таким образом, при использовании кодирующего диска с 1024 отверстиями можно получить возможность измерения 4096 ступеней скорости.

Состояниям счетчика удобно поставить в соответствие некоторый цифровой код, формируемый уровнями сигналов, снимаемых с датчиков (S1 и S2). Соответствие состояния датчика и цифрового кода показано на рис.__.

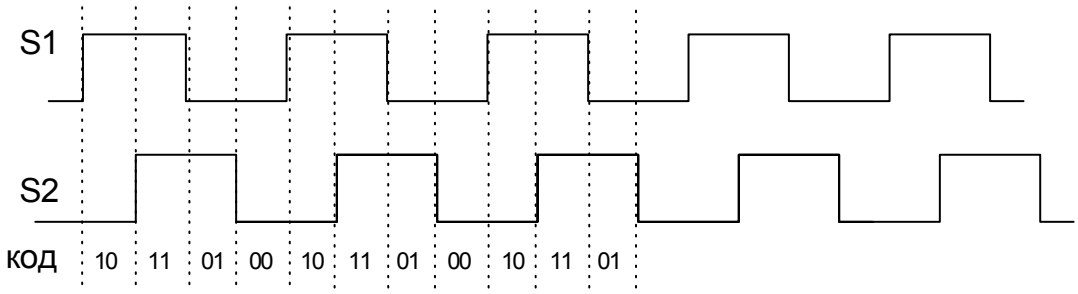


Рис. Иллюстрация последовательности смены кодов на выходах линий S_1 и S_2.

На рис.___ приведен листинг процедуры на языке ассемблера, реализующей формирование цифрового кода угла поворота вала датчика с учетом направления вращения кодового диска.

Код угла формируется в регистре r2.

Регистры r1, r3 текущего банка регистров используются в качестве рабочих регистров для хранения предыдущего кода состояния датчика и промежуточных результатов.

Для реализации этой процедуры необходимо выходы ФИД (линии S1, S2, S3) подключить к соответствующим выводам порта P0 микроконтроллера.

```
org   0000h
Start: mov  r1,   0                ;
       mov  r2,   0                ;
       mov  r3,   0                ;
;      r1
;      r2
go:    mov  a,    r3               ;
       mov  r1,   a                ;
       mov  a,    90h              ;
       mov  r3,   a                ;
       xrl  a,    r1               ;
       cjne a,    0,    m_clear    ;
       jmp  go
;
m_clear:  xrl  a, 00000100b        ;
       cjne a,    0,    m00        ;
       clr  r2                     ;
;
m00:  cjne r3,   00b,  m01         ;
      cjne r1,   01b,  m_i         ;
      jmp  m_d                     ;

m01:  cjne r3,   01b,  m11         ;
      cjne r1,   11b,  m_i         ;
      jmp  m_d                     ;

m11:  cjne r3,   11b,  m10         ;
      cjne r1,   10b,  m_i         ;
      jmp  m_d                     ;

m10:  cjne r1,   00b,  m_i         ;
m_d:  dec  r2                      ;
       jmp  go                     ;
m_i:  inc  r2                      ;
       jmp  go
```

Рис. Листинг программы, формирующей цифровой код по сигналам ФИД.

III. ОБЪЕКТ ИССЛЕДОВАНИЯ, ПРИБОРЫ, ОБОРУДОВАНИЕ

В работе используется специализированный стенд, разработанный на базе микроЭВМ МК51, осциллограф, персональный компьютер. Для сопряжения микроЭВМ и персонального компьютера используется специальный шлейф, подключаемый к стандартному параллельному порту персонального компьютера.

IV. ЗАДАНИЕ НА РАБОТУ

Изучить принцип работы ФИД. Рассмотреть варианты формирования цифрового кода с использованием ФИД. Изучить программу формирования цифрового кода, приведенную на рис.___. По приведенной программе нарисовать алгоритм формирования цифрового кода. Ввести программу в память ПК, провести отладку и тестирование программы с использованием лабораторного стенда. Просмотреть диаграммы сигналов на выводах ФИД с помощью осциллографа.

V. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

24. Изучить функциональную схему микроконтроллера.
25. По принципиальной схеме контроллера определить наименования портов, используемых для обмена информацией.
26. Изучить названия линий управления, используемых для управления обменом информацией внутри микроконтроллера.
27. Изучить способ формирования сигналов управления для обмена информацией микроконтроллера с персональным компьютером и внешними устройствами.
28. Изучить режимы работы микроЭВМ.
29. Составить диаграммы сигналов, соответствующих режимам обмена информацией с модулями УСО и микроЭВМ.
30. По указанию преподавателя составить функциональную схему модуля ввода сигналов с клавиатуры 4*4 или вывода сигналов на ЖКИ.
31. Составить алгоритм обмена информацией персонального компьютера и микроЭВМ при помощи рассмотренного эмулятора.

VI. ОФОРМЛЕНИЕ ОТЧЕТА

В отчете должны быть представлены алгоритмы программ, листинги программ. Функциональная схема подключения осциллографа к микроЭВМ, диаграмма сигнала, полученная с помощью осциллографа.

VII. КОНТРОЛЬНЫЕ ВОПРОСЫ

VIII. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

7. Горячев О.В. Микропроцессоры и микропроцессорные вычислительные системы. – Тула, ТулГУ, 1998.
8. Микропроцессорные автоматические системы регулирования. Основы теории и элементы./Под ред. В.В.Солодовникова,- М., 1991.
9. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат. 1990г. - 224с.

Лабораторная работа № 5. Цифровая система управления приборного электрического следящего привода постоянного тока. Изучение функциональной схемы УСО и принципиальных электрических схем.

Цель работы:

1. Изучение устройства и принципов работы цифрового электрического привода (ЦЭСП)
2. Экспериментальное исследование процессов, протекающих в ЦЭСП.

1. Техническое описание

Краткая характеристика лабораторного стенда.

Цифровой электрический следящий привод постоянного тока с управлением от персонального компьютера (ПК) состоит из двух основных частей: аппаратной и программной. Конструктивно ЦЭСП представляет собой следующие основные блоки: двигатель и редуктор, усилитель мощности, блок питания, интерфейсный модуль и устройство управления (ПК). Укрупненная структурная схема такого привода приведена на рисунке 1. Нагрузкой привода является его механическая часть: редуктор и набор датчиков.

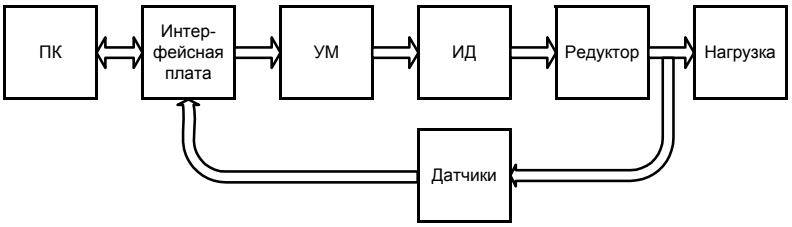


Рисунок 1. Структурная схема ЦЭСП.

Персональный компьютер вырабатывает необходимые сигналы управления, а также выполняет функцию корректирующего устройства. Программное обеспечение позволяет настраивать различные параметры ЦЭСП, такие как такт квантования, коэффициент передачи и т.д. Также сразу можно оценить влияние тех или иных параметров на качество переходного процесса.

Имеется возможность использовать различные типовые управляющие воздействия.

Информация в ПК представлена в цифровом виде, а управление происходит непрерывным объектом, следовательно, необходимо сопрягающее устройство – интерфейсная плата. Преобразование цифрового управляющего воздействия в аналоговый вид, в широтно-модулированный сигнал, преобразование сигналов с аналоговых датчиков скорости и угла в цифровой вид, а также преобразование сигналов с фотоимпульсного датчика – все это позволяет производить интерфейсная плата. Она содержит ряд дополнительных функциональных возможностей, таких как настройка частоты ШИМ, АЦП и ЦАП, а также выбор входного канала АЦП (используется при мультиплексировании сигналов с разных датчиков).

Описание аппаратного обеспечения.

В качестве силового преобразователя используется импульсный усилитель мощности с симметричным ШИМ-управлением. В качестве датчика угла используется вращающийся трансформатор (ВТ), включенный по схеме ЛВТ. Для сравнения и получения более высокой точности и помехозащищенности используется фотоимпульсный датчик (ФИД) угла. Используя ФИД можно легко получить информацию об угловой скорости.

Для получения кода угла с помощью вращающегося трансформатора необходимы дополнительные элементы – генератор, демодулятор и фильтр низких частот. Эти элементы расположены в отдельном корпусе, также в этом корпусе располагается импульсный усилитель.

Рассмотрим функциональную схему интерфейсной платы.

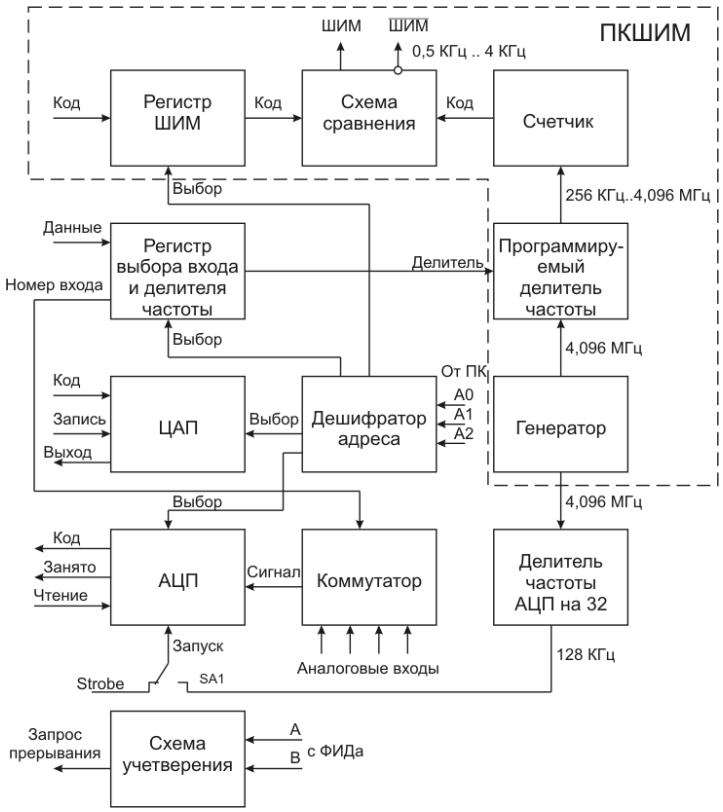


Рисунок 2. Функциональная схема модуля ввода вывода.

Аппаратно модуль состоит из следующих основных компонентов:

Коммутатор аналоговых сигналов подключает к АЦП один из аналоговых входов.

АЦП преобразует аналоговый сигнал, поступающий с коммутатора в цифровой код.

ЦАП преобразует цифровой код, формируемый в компьютере, в аналоговый сигнал.

Регистр ШИМ хранит информацию о скважности. Код 80h соответствует скважности $\gamma = 0.5$; код FFh соответствует $\gamma = 1$.

Счетчик ШИМ увеличивает свое значение от 0 до FFh за один период ШИМ-сигнала.

Схема сравнения формирует логическую единицу, если число в регистре ШИМ больше, чем число на выходе счетчика, иначе – логический ноль.

ФИД и схема учетверения счетных импульсов

Схема выделения фронтов и срезов импульсов с фотоимпульсного датчика собрана на двух элементах XOR. Сигналы А и Б представляют собой прямоугольные импульсы, сдвинуты относительно друг друга на 90 градусов (смотри диаграмму на рисунке 3). После первого элемента XOR частота следования импульсов удваивается (сигнал КТ1). Далее сигнал КТ1 инвертируется и задерживается на несколько микросекунд (элементы DD3, DD2 и RC-цепь). В результате сигналы КТ1 и КТ2 поступают на второй элемент XOR, на выходе которого формируются импульсы, срез которых совпадает с фронтами и срезами импульсов канала А и В. В данном приводе эти импульсы используются для формирования сигнала запроса прерывания. Элементы цепи задержки DD3, R, C, DD2 определяют длительность импульса запроса прерывания IRQ.

По каждому прерыванию происходит считывание каналов А и В в компьютер. Допустим предыдущее состояние канала А=1, а канала В=0 (верхний кружок на рисунке 4). При повороте происходит смена состояния А=1, В=1 (поворот в право) или А=0, В=0 (при повороте в лево). Таким образом, сравнивая предыдущее состояние каналов А и В с текущим состоянем можно сделать вывод о направлении вращения вала ФИД. На рисунке 4 внешними стрелками показаны переходы состояний, соответствующие вращению в право, а внутренними – вращение в лево. В компьютере есть переменная, которая хранит значение угла в виде числа. В обработчике прерывания анализируется направление движения, и если произошло движение в право, то эта переменная инкрементируется, иначе – декрементируется.

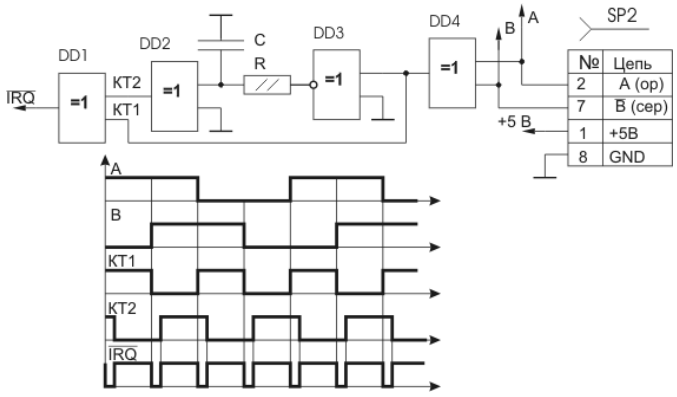


Рисунок 3. Схема учетверения и диаграмма,
поясняющая ее работу.

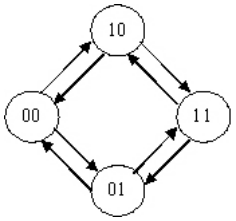


Рисунок 4. Диаграмма состояний фотоимпульсного датчика.

Алгоритм работы программного обеспечения

В программе используется кольцевой буфер – массив из 10000 элементов. Существует глобальная переменная index которая содержит номер текущего элемента массива. Элементом массива является вектор, состоящий из задающего воздействия (угол), выходного угла, скорости и скважности. Заполнение массива происходит в процедуре QuantControl. Процедура вызывается в строго определенные моменты времени, соответствующие такту квантования.

В начале каждого такта квантования происходит обращение к драйверу, который возвращает текущее значение угла и скорости. Эти значения записываются в массив, после чего вызывается функция, которая формирует сигнал скважности. Значение скважности при помощи драйвера записывается в порт ЦАП и в порт ШИМ, расположенные на интерфейсной плате.

Изображение осциллографа обновляется приблизительно каждый сотый раз (этот параметр можно настраивать). В конце каждого такта определяется, нужно ли обновить картинку. Если нужно, то основной форме посылается сообщение, обработчик которого прорисовывает картинку.

```
procedure QuantControl(uTimerID, uMessage: UINT;
   dwUser, dw1, dw2: DWORD) stdcall;
var Udv: byte;
begin
  inc(Count); inc(index);                    //Инкрементируем текущий индекс
  if index>MaxData then index:=0;            //При необходимости его закольцовываем
  GetDataVec;                                        //Получаем значение угла и скорости
  Mas[0,index]:=DV.Angle*1.53398e-3;     // pi/2048 = 1.53398e-3
```

```
  Mas[2,index]:=Form1.GetSpd;                    //Записываем текущее значение скорости
  Udv:=Form1.Correction;                  //Получаем значение скважности
  Mas[3,index]:=Udv/255;                  //и записываем его в массив
  SetShim_DAC(Udv);                              //Записываем скважность в порт
  if ((Count*Signal1.TQuant) mod Scope1.RefTime)=0 then
    PostMessage(Form1.Handle,WM_USER+100,0,Count);    // Обновление осциллографа
end;
```

Ниже приведен фрагмент процедуры обработки прерывания, которая является частью виртуального драйвера:

```
        mov dx, LptStatus               ;Записываем в dx адрес порта Status (параллельный порт)
        in al, dx                       ;Читаем значение порта
        mov ah, bit4                    ;ФИД подключен к LPT, значение сигнала A и B содержится в 4 и 5
битах
        or ah, bit5                     ;Далее происходит выделение именно этих бит
        and al, ah                      ;al:=status and (bit4 or bit5);
        shr al, 4                       ;al:=xxxxxx45b
        mov ah, [oldStatus]             ;B ah находится предыдущее значение сигналов A и B, а в al - текущее
        .IF ((al==0)&&(ah==1)) || ((al==1)&&(ah==3)) || ((al==3)&&(ah==2)) || ((al==2)&&(ah==0))
                dec [esi].Angle         ;Определяется направление вращения с соответствии с рисунком 4
        .ELSE
                inc [esi].Angle
        .ENDIF                          ;Для упрощения кода был использован макрос .IF
        mov [oldStatus], al             ;Сохраняем значение сигналов A и B в переменную oldStatus
```

Определение скорости происходит несколько сложнее. Смысл заключается в том, что подсчитывается промежуток времени между тремя прерываниями. Значение скорости – величина обратная этому промежутку времени. Если прерывания происходят редко (двигатель медленно вращается), то скорость получается не гладкой (рваный график). Для сглаживания сигнал скорости пропускают через цифровой фильтр низких частот. В результате получается приемлемый сигнал.

Интерфейс программного обеспечения ЦЭСП на базе ПК (приложение Scope)

Физически программное обеспечение состоит из двух частей:

- Специальный виртуальный драйвер sample.vxd. Драйвер обеспечивает доступ к портам ввода-вывода и осуществляет перехват прерывания от параллельного порта ПК. В обработчике прерывания вычисляется код угла и значение скорости вращения.

- Приложение Scope.exe, которое является основной частью программного обеспечения.

На рисунке 5 представлено главное окно программы. Основную часть окна занимает осциллограф.

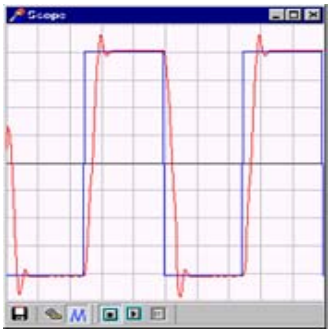


Рисунок 5. Главное окно программы

В нижней части окна располагается панель управления. Слева направо расположены кнопки:

1. «Просмотр» - позволяет более подробно просмотреть результат работы стенда.
2. «Настройки» - вызывает окно настроек. Оно состоит из четырех закладок:

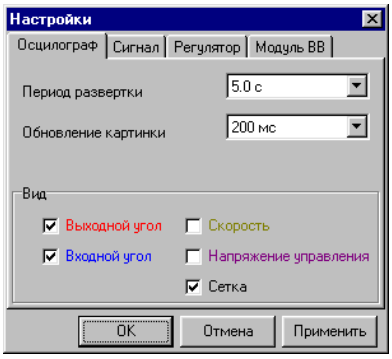


Рисунок 6. Закладка «Осциллограф».

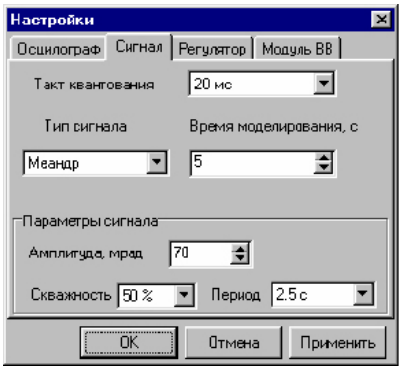


Рисунок 7. Закладка «Сигнал».

- Закладка «Осциллограф» (рисунок 6). Позволяет настроить длительность развертки осциллографа и период обновления картинки.
- Закладка «Сигнал» (рисунок 7). Позволяет выбрать такт квантования в системе; изменить тип входного сигнала, а также настроить параметры этого сигнала.
- Закладка «Регулятор» (рисунок 8). Позволяет выбрать тип регулятора.
- Закладка «Модуль ВВ» (рисунок 9). Здесь можно провести диагностику регистров модуля ввода-вывода, изменить частоту ШИМ, а также выбрать входной канал АЦП.

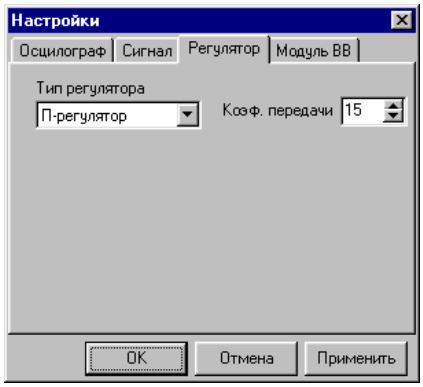


Рисунок 8. Закладка «Регулятор».

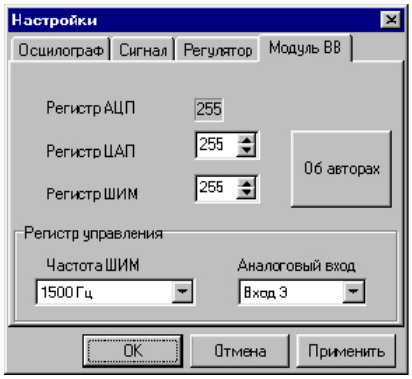


Рисунок 9. Закладка «Модуль ВВ».

3. «Автомасштаб» - автоматически масштабирует изображение.
4. «Споп» - кнопка останавливает процесс управления.

5. «Пуск» - включает процесс управления.

6. «Пауза» - приостанавливает процесс управления. Используется при сохранении данных или при вызове настроек.

2. Приборы и оборудование

- Персональный компьютер с приложением Scope.exe;
- Двухлучевой осциллограф;
- Цифровой следящий привод постоянного тока.

3. Порядок выполнения работы.

- Включить осциллограф. Включить блок питания привода. Включить ПК.
- Запустить приложение Scope.exe.
- Включить тумблер +27 В на панели блока «Усилитель, генератор, демодулятор», при этом должен начать вращаться ротор двигателя.
- Подключить осциллограф к гнезду «ШИМ» на панели блока «Модуль ввода вывода».
- Вызвать окно «Настройки» и выбрать закладку «Модуль ВВ»
- Записать в регистр ШИМ несколько различных значений, например, 64, 128, 192. Занести показания осциллографа в отчет, причем на экране должно умещаться 2 периода сигнала.
- Записать в регистр ШИМ 128 и установить несколько различных частот ШИМ, например, 1500 Гц, 2000 Гц, 2600 Гц. Занести показания осциллографа в отчет, причем период развертки осциллографа должен оставаться неизменным.
- Подключить осциллограф к гнезду «ЦАП» на панели блока «Модуль ввода вывода».
- Записать в регистр ЦАП несколько различных значений, например, 64, 128, 192. Замерить соответствующие напряжения и занести их в таблицу.

- Подключить осциллограф к каналу A и B на панели блока «Модуль ввода вывода». Записать в регистр ШИМ 255, 192, 0. Сделать выводы о длительности импульсов.
- Подключить осциллограф к каналу IRQ и B на панели блока «Модуль ввода вывода». Записать в регистр ШИМ 255. Занести результаты в отчет.
- На блоке «Усилитель, генератор, демодулятор» отключить питание двигателя + 27 В. В программе выбрать закладку «Осциллограф»; установить период развертки 500 мс; в группе «Вид» отключить всё кроме скважности. На закладке «Сигнал» установить такт квантования 5 мс, тип сигнала – синус, время моделирования – 2000 с, амплитуду сигнала – 40 мрад, период – 250 мс.
- Подключить осциллограф к гнезду «ЦАП». Нажать кнопку «Пуск» главного окна программы. Занести в отчет результаты работы программы и показания осциллографа.
- Выбрать закладку «Сигнал»; установить такт квантования 20 мс. Сравнить полученные эпюры с эпюрами на 5 мс. Занести показания осциллографа в отчет.
- Выбрать закладку «Сигнал»; установить такт квантования 10 мс. Установить тип сигнала пила «Пила». Занести в отчет результаты работы программы.
- Выбрать закладку «Сигнал»; установить амплитуду 60 мрад. Сравнить полученные эпюры с эпюрами на 40 мрад. Занести в отчет результаты работы программы.
- Подключить осциллограф к гнезду «ШИМ».
- Выбрать закладку «Осциллограф» и установить период развертки 5.0 с. Выбрать закладку «Сигнал»; тип сигнала – синус, амплитуда – 20 мрад, период – 2.0 с. Получить на осциллографе изображение 2-х периодов. Увеличивать амплитуду сигнала с 20 мрад до появления нелинейных искажений с шагом 10 мрад. Сделать выводы о глубине модуляции ШИМ-сигнала.
- Закрыть программу. Выключить блок питания привода.

4. Содержание отчета

- Название и номер работы, цель работы;
- Все эпюры с пояснениями, указанные в порядке выполнения работы;
- Выводы.

5. Контрольные вопросы

- Что такое скважность? Какое число в регистре ШИМ соответствует скважности 0.5, 0.75, 1?
- Чем объяснить звук, издаваемый двигателем на скважности 0.5?
- Объясните принцип выделения фронтов и срезов сигналов с ФИД.
- Каким образом получается значение угла в компьютере? Как определить направление счета?
- Как влияет такт квантования на качество формируемого сигнала?
- Что такое нелинейные искажения? От чего зависит возникновение нелинейных искажений в данной системе?
- Что такое глубина модуляции? От чего она зависит?

6. Список литературы

1. Гук М. Аппаратные средства IBM PC. – С-пб.: Питер, 1996. – 212 с.
2. Пухальский Г. И. Новосельцева Т. Я. Проектирование дискретных устройств на интегральных микросхемах. Справочник. – М.: Радио и связь, 1990.- 304 с.
3. Старшин В. В., Урусов А.В. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990. – 219 с.
4. Следящие приводы: В 3т. 2-е изд., доп. и перераб. /Под ред. Б. К. Чемоданова. Т. I: Теория и проектирование следящих приводов /Е. С. Блейз, А. В. Зимин, Е. С. Иванов и др. – М.: Изд-во МГТУ им. Н. Э. Баумана, 1999. – 904 с.

Лабораторная работа № 6. Цифровая система управления приборного электрического следящего привода постоянного тока. Исследование характеристик в среде SimuLink.

Цель работы:

1. Знакомство с пакетом SimuLink, изучение последовательности построения математической модели линейной системы в среде системы SimuLink, расчет переходных процессов в системе.
2. Исследование процессов, протекающих в цифровых следящих системах, при помощи пакета моделирования Simulink.
3. Получение основных характеристик модели ЦЭСП.

1. Теоритические сведения

Общие сведения о пакете SimuLink.

Пакет SimuLink позволяет осуществлять исследование (моделирование) поведения динамических линейных и нелинейных систем. Ввод характеристик систем производится в диалоговом режиме, путем графической сборки схемы соединений элементарных стандартных звеньев. В результате такой сборки образуется модель исследуемой системы, которую в дальнейшем будем называть S- моделью. Модель хранится в файле с расширением .mdl.

Создание моделей в пакете SimuLink основано на использовании технологии Drag-and-Drop (тащи и отпускай). В качестве «кирпичиков» для построения S- модели применяются модули (или блоки), хранящиеся в библиотеке SimuLink. Любая S- модель может иметь иерархическую структуру, т.е. состоять из моделей более низкого уровня, причем число уровней иерархии практически не ограничено. В ходе моделирования имеется возможность наблюдать за процессами, происходящими в системе. Для этого используются специальные смотровые окна, входящие в библиотеку SimuLink.

Состав библиотеки SimuLink может быть пополнен пользователем за счет разработки собственных блоков.

Запуск пакета SimuLink можно произвести из командного окна MatLab, например, при помощи соответствующей пиктограммы в панели инструментов (рис.1).

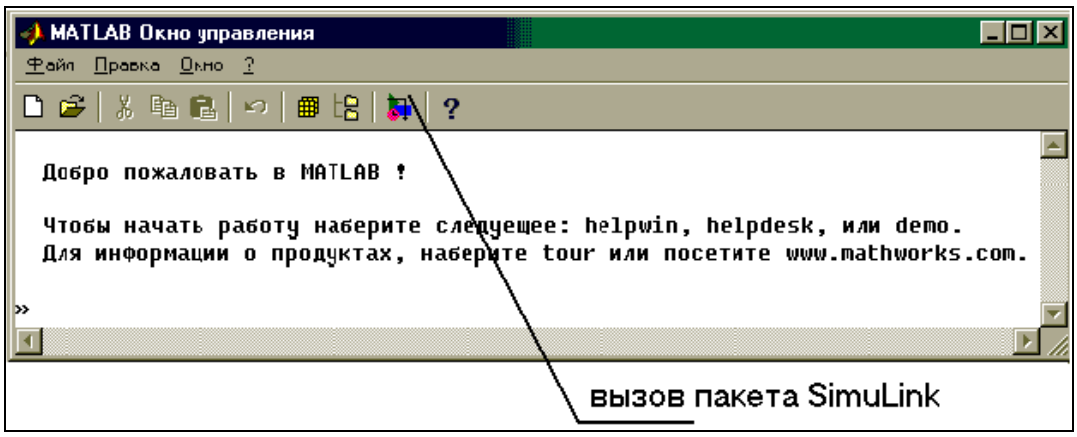


Рис.1.

При запуске SimuLink открываются два окна (рис.2):

- окно Library: simuLink – с перечнем основных разделов библиотеки SimuLink;
- пустое окно untitled (заготовка для создания новой S-модели, MDL-файла, или схемного изображения моделируемой системы).

-

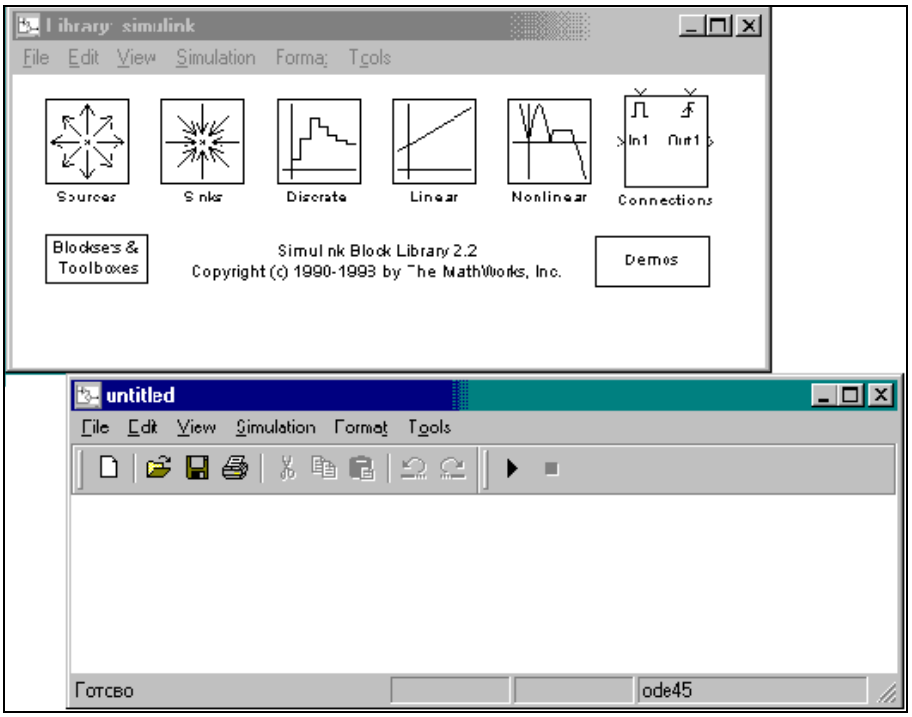


Рис.2 .

Оба окна имеют сходную структуру и содержат строку меню, панель инструментов и рабочее поле.

Библиотека модулей (блоков) SimuLink представляет собой набор визуальных объектов, являющихся статическими или динамическими звеньями систем управления, источниками или приемниками сигналов и т.д. При помощи модулей SimuLink можно составлять блок схему любого устройства, путем

переноса их в рабочее поле и соединения между собой функциональными связями.

Например, структурной схеме линейной САУ, отображенной на рис.3 соответствует модель в среде SimuLink (S- модель), представленная на рис.4.

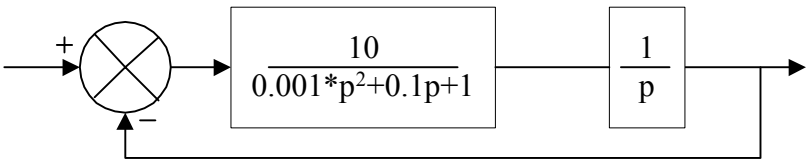


Рис.3.

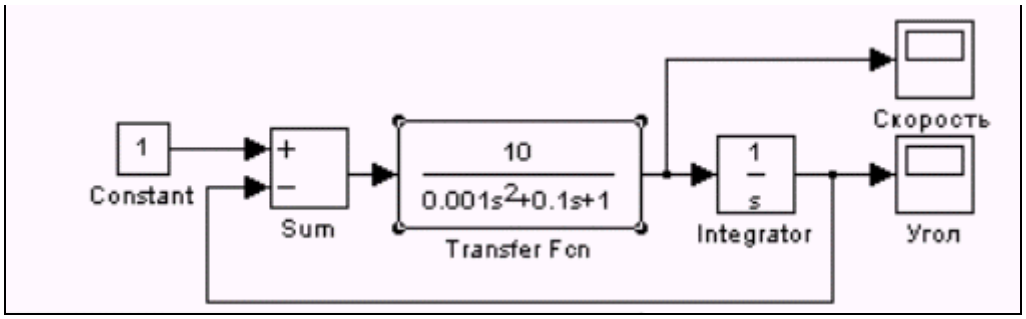


Рис.4.

Библиотека блоков разбита на семь разделов. Шесть из них являются основными и не могут изменяться пользователем:

- **Sources** – источники;
- **Sinks** – приемники;
- **Discrete** – дискретные элементы;
- **Linear** – линейные элементы;
- **Nonlinear** – нелинейные элементы;
- **Connections** – связи (соединения).

Седьмой раздел **Blockets & Toolboxes** (наборы блоков и инструменты) – содержит блоки, созданные пользователем и включенные в рабочую конфигурацию пакета.

Общий вид блоков библиотеки **Library: simuLink** показано на рис.2. Каждый блок библиотеки раскрывается двойным щелчком правой кнопки «мышки» после установки ее указателя на соответствующий блок. Раскрытое поле блоков **Sources** (источники) показано на рис.5.

Аналогично раскрываются поля и других разделов библиотеки **Library: simuLink.** Общий вид полей **Sinks, Discrete** и **Linear** показаны на рис.6 а,б и в соответственно.

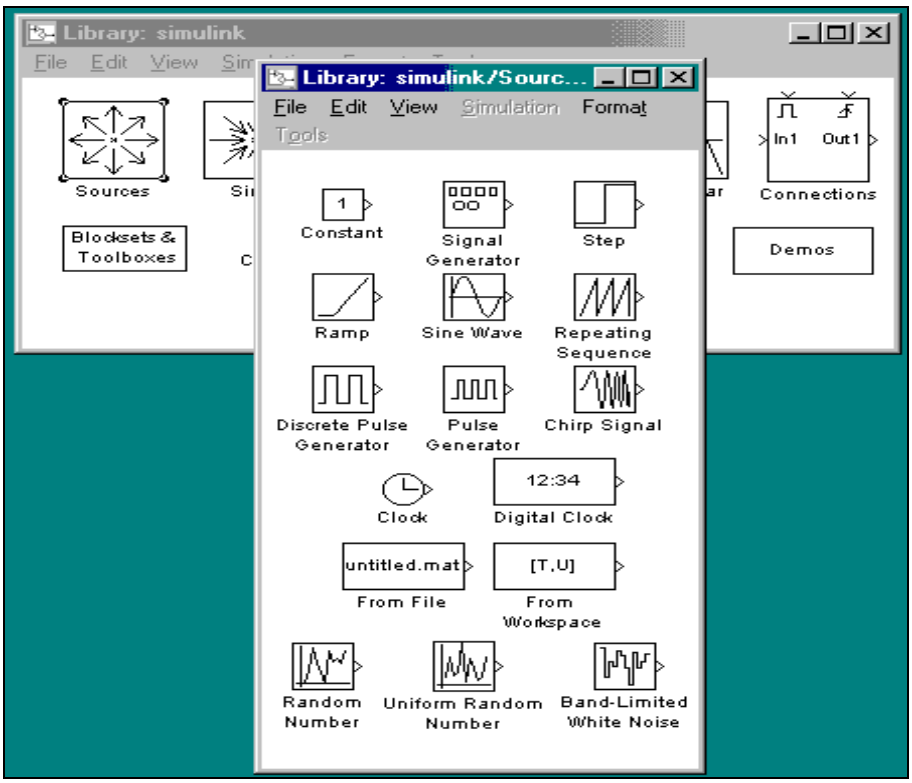
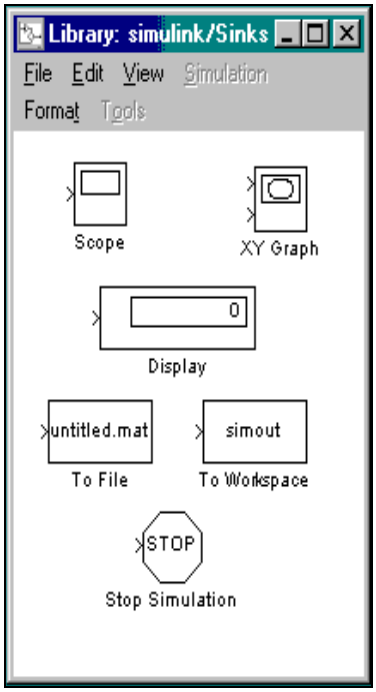
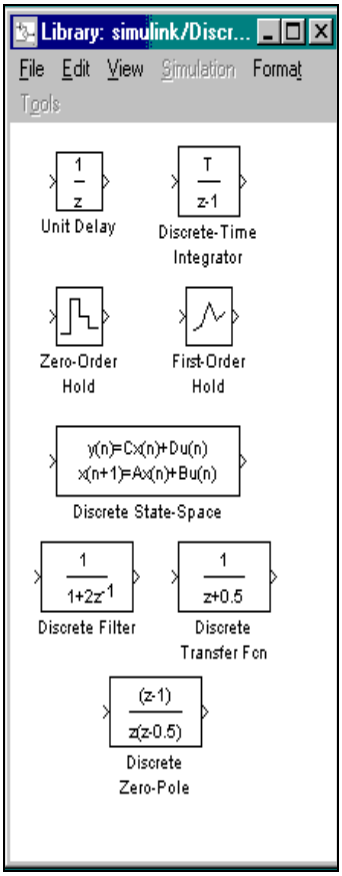


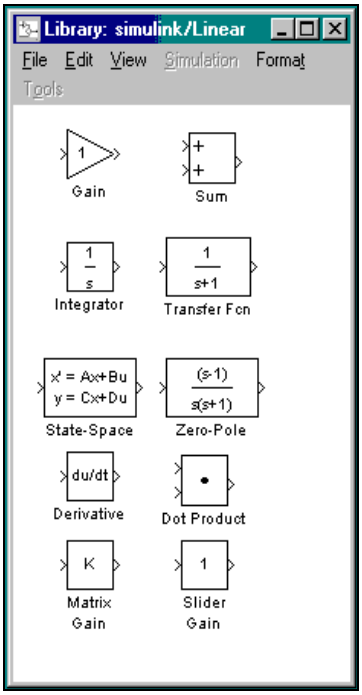
Рис.5.



а)



б)



в)

Рис.6.

При выполнении лабораторных работ по потребуются следующие блоки из библиотеки **Library: simuLink**:

Раздел **Sources:**

- **Constant –** ступенчатый сигнал с устанавливаемой амплитудой ;
- **Sine Wave –** гармонический сигнал с задаваемыми параметрами.

Раздел **Sinks:**

- **Scope –** модуль «осциллограф» используется для отображения результатов расчетов.

Раздел **Linear:**

- **Gain** – линейное передаточное звено;
- **Transfer Fcn –** определение передаточной функции динамического звена;
- **Integrator –** идеальное интегрирующее звено;
- **Sum –** звено-сумматор.

Раздел **Discrete:**

- **Zero-Order Hold –** экстраполятор нулевого порядка.

Каждое звено переносится на рабочее поле путем выполнения следующих операций:

- раскрытия соответствующего раздела библиотеки;
- установки указателя «мышки» на требуемый модуль;
- захват модуля осуществляется нажатием на правую кнопки «мышки»;
- перенос модуля на рабочее поле и фиксация его путем отпускания правой кнопки «мышки»;
- коррекция параметров модуля.

Коррекция параметров модуля осуществляется путем двойного щелчка на правую кнопку «мышки» после установки указателя на соответствующий модуль. При этом раскрывается поле параметров модуля, в котором и необходимо произвести соответствующие изменения.

После размещения на наборном поле необходимых модулей и установки параметров, осуществляется соединение входов и выходов модулей в соответствии со схемой моделируемой системы.

Запуск процесса расчета динамического состояния модели осуществляется выбором в меню **Simulation** команды **Start.** Для просмотра результатов моделирования необходимо предварительно ввести в модель модули **Scope.** В процессе вычисления окна модулей **Scope** раскрываются также двойным щелчком правой кнопки «мышки» после установки ее указателя на соответствующий модуль. Пример наборного поля со схемой

модели системы, заданной своей структурной схемой (рис.3) и раскрытыми окнами модулей **Scope** приведены на рис.7.

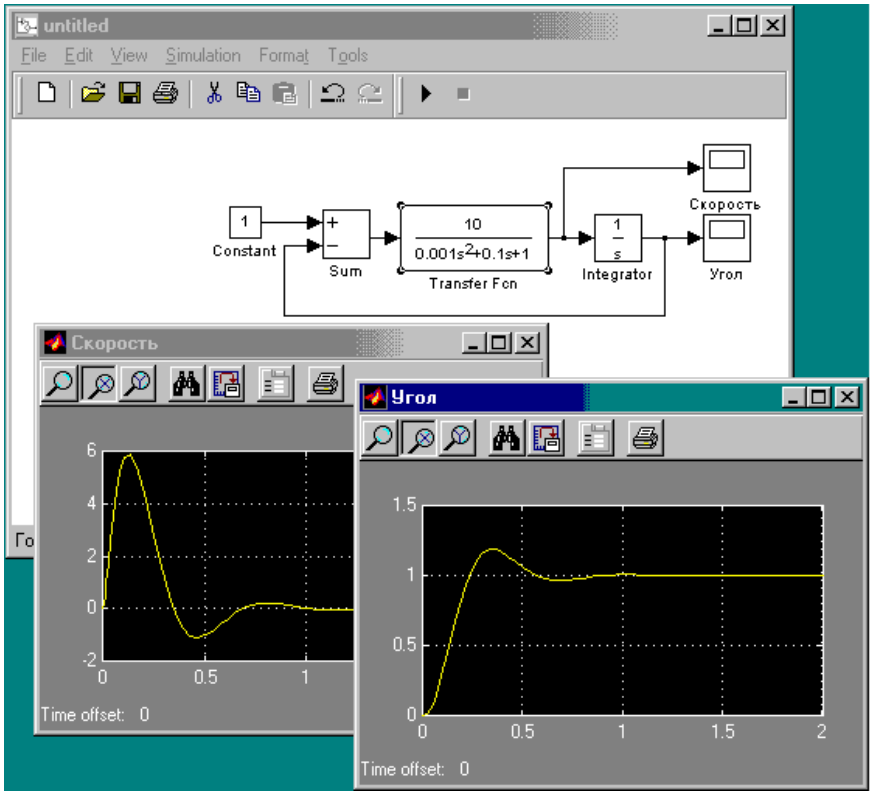


Рис.7.

Выход из среды SimuLink осуществляется путем закрытия главного окна пакета MatLab.

Сохранение модели выполняется путем выбора команды Save as… в меню File. При этом необходимо предварительно установить папку для сохранения S-моделей.

Описание модели

Структурная схема ЦЭСП приведена на рисунке 8. Голубым цветом указаны элементы, которые реализованы программно внутри ПК. Малиновым цветом показан формирователь ШИМ-сигнала, расположенный на плате модуля ввода вывода (смотри описание стенда лаб. раб. №1). Красным цветом показан усилитель мощности, зеленым – исполнительный двигатель, серым – редуктор. Желтым цветом показан фотоимпульсный датчик, работающий в режиме датчика угла, поэтому на схеме изображен интегратор, который в реальной системе не существует.

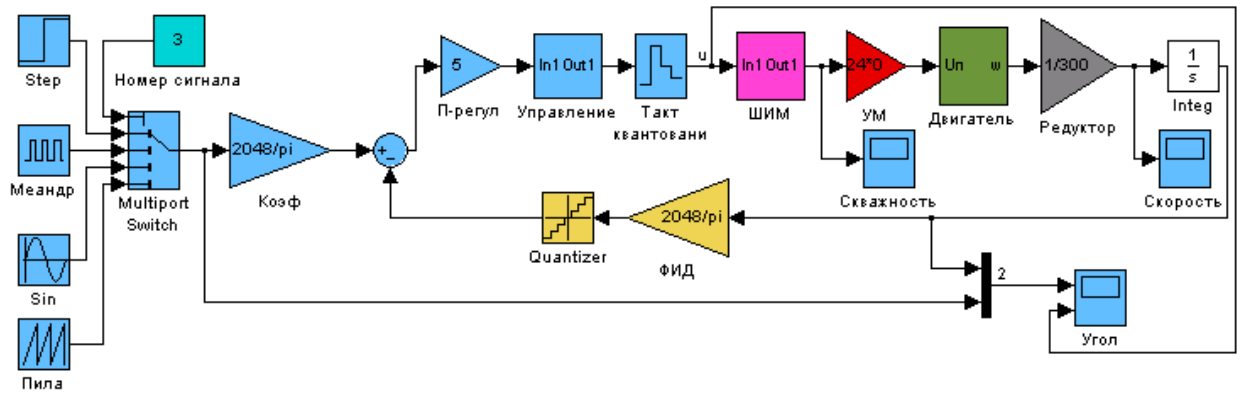


Рис. 8. Структурная схема модели ЦЭСП

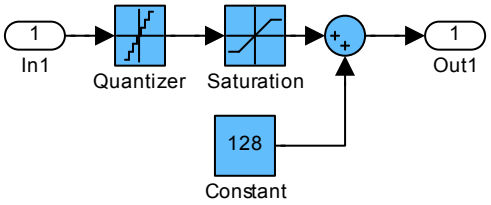


Рис. 9. Подсистема формирования скважности в виде цифрового кода.

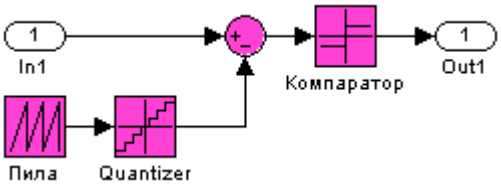


Рис. 10. Подсистема формирования скважности в виде ШИМ-сигнала.

Исполнительный двигатель представлен моделью 3-го порядка (смотри рис. 11).

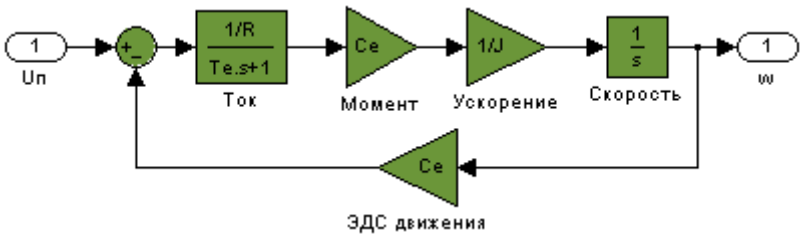


Рисунок 11. Подсистема двигателя постоянного тока.

2. Приборы и оборудование

- Персональный компьютер с установленной системой Matlab;

3. Порядок выполнения работы.

➢ Реализовать модель, представленную на рис. 3 в среде SimuLink

- Запустить Simulink и открыть файл Analiz.mdl
- Установить коэффициент передачи УМ в ноль, установить такт квантования 5 мс, тип входного сигнала – синус (номер 3 Multiport Switch), амплитуду сигнала – 40 мрад. Занести в отчет результаты Scope.
- Установить такт квантования 20 мс. Сравнить полученные эпюры с эпюрами на 5 мс. Занести показания Scope отчет.
- Установить такт квантования 10 мс. Установить тип сигнала пила «Пила», амплитуду сигнала – 40 мрад. Занести в отчет переходный процесс по углу.
- Установить амплитуду 60 мрад. Сравнить полученные эпюры с эпюрами на 40 мрад. Занести в отчет переходный процесс по углу.
- Установить коэффициент передачи УМ равным 24.
- Выбрать тип сигнала 1(t) и установить амплитуду 40 мрад. Запустить систему и занести в отчет вид переходного процесса и показатели качества. Установить такт квантования 20 мс, 50 мс, 100 мс и занести результаты в отчет.
- Установить такт квантования 10 мс. Получить переходный процесс для нескольких значений амплитуды входного сигнала (40 мрад, 150 мрад, 300 мрад). Посчитать для каждого графика показатели качества. Занести результаты в отчет.
- Установить амплитуду 150 мрад. Установить коэф. передачи П-регулятора равным 1. Занести результаты в отчет. Найти такой коэф. предачи, при котором система становится неустойчива. Сохранить коэффициент $К_{гр}$ и вид переходного процесса в отчет.
- Установить коэф. передачи 200. Измерить частоту и амплитуду автоколебаний при разных тактах квантования (5 мс, 10 мс, 20 мс, 50 мс). Сохранить переходный процесс по углу в отчет.
- Установить коэф. передачи 5, такт квантования 10 мс, тип сигнала - синус с частотой 8 рад/с, 4 рад/с. Вычислить временное и фазовое запаздывание.

- Установить тип сигнала – меандр, амплитуда – 60 мрад. Сохранить переходный процесс по углу и по скорости. Установить амплитуду - 250 мрад, 500 мрад. Сохранить переходный процесс по углу и по скорости.

4. Содержание отчета

- Название и номер работы, цель работы;
- Все эпюры с пояснениями, указанные в порядке выполнения работы;
- Выводы.

5. Контрольные вопросы

- Какие основные показатели качества вы знаете?
- Что такое такт квантования? Как такт квантования влияет на основные показатели качества?
- На что влияет коэф. передачи регулятора?
- Как в системе с ограничением могут возникнуть автоколебания? Как влияет такт квантования на амплитуду и частоту автоколебаний?
- Чем объяснить зависимость перерегулирования от величины входного воздействия?

Список литературы

1. Справочное руководство по системе Simulink.

2. Лазарев Ю.Ф. MatLAB 5.x. –К.: Издательская группа BHV,2000 – 384с. (Серия «Библиотека студента»).

3. Конспект лекций по линейной теории автоматического управления.

4. Конспект лекций по нелинейной теории автоматического управления.

5. Следящие приводы: В 3т. 2-е изд., доп. и перераб. /Под ред. Б. К. Чемоданова. Т. I: Теория и проектирование следящих приводов /Е. С. Блейз, А. В. Зимин, Е. С. Иванов и др. – М.: Изд-во МГТУ им. Н. Э. Баумана, 1999. – 904 с.

Лабораторная работа № 7. Цифровая система управления приборного электрического следящего привода постоянного тока. Сравнительный анализ экспериментальных и расчетных характеристик.

Цель работы:

1. Практическое исследование явлений, возникающих в цифровых следящих системах.

2. Получение основных характеристик ЦЭСП.

1. Теоретическое описание

В линейной теории автоматического управления рассматриваются идеализированные линейные объекты. Характеристики таких объектов зависят только от собственных параметров объекта (коэффициенты передачи и постоянные времени).

В реальных системах с цифровым управлением появляются такие особенности, как ограничения уровней сигналов, дискретность уровней сигналов с датчиков, дискретность по времени управляющих воздействий на объект управления.

В данной лабораторной работе целенаправлено задаются такие входные сигналы и устанавливаются такие коэффициенты передачи, которые позволяют выявить поведение цифровой системы в «нестандартных» условиях работы. В результате можно оценить:

- влияние такта квантования на показатели качества;
- влияние коэффициента передачи в системе с ограничением;
- влияние такта квантования на частоту и амплитуду дискретной релейной системы;
- зависимость переходных процессов от величины входного воздействия в системе с ограничением.

2. Приборы и оборудование

- Персональный компьютер с приложением Scope.exe;
- Цифровой следящий привод.

3. Порядок выполнения работы.

- Включить блок питания привода, включить ПК.
- Запустить приложение Scope.exe.
- Включить тумблер +27 В на панели блока «Усилитель, генератор, демодулятор».
- Выбрать закладку «Сигнал» и установить такт квантования 10 мс, амплитуду сигнала – 40 мрад. Запустить систему. Вызвать окно «Просмотр» и занести в отчет вид переходного процесса и показатели качества. Установить такт квантования 20 мс, 50 мс, 100 мс и занести результаты в отчет.
- Выбрать закладку «Сигнал» и установить такт квантования 10 мс. На закладке «Регулятор» установить коэф. передачи 1. Занести результаты в отчет. Найти такой коэф. предачи, при котором система становится неустойчива. Сохранить вид переходного процесса в отчет.
- Установить коэф. передачи 200. Измерить частоту и амплитуду автоколебаний при разных тактах квантования. Сохранить результаты в отчет.
- Установить коэф. передачи 5. На закладке «Сигнал» установить такт квантования 10 мс, амплитуду 70 мрад, тип сигнала - синус, период 750 мс, 1.5 с. Вычислить временное и фазовое запаздывание.
- Установить тип сигнала – меандр, амплитуда – 60 мрад, период – 1.0 с. Сохранить переходный процесс по углу и по скорости. Установить амплитуду - 250 мрад, 500 мрад. Сохранить переходный процесс по углу и по скорости.
- Закрыть программу. Выключить двигатель. Выключить блок питания привода.

4. Содержание отчета

- Название и номер работы, цель работы;
- Все эпюры с пояснениями, указанные в порядке выполнения работы;
- Выводы.

5. Контрольные вопросы

- Какие основные показатели качества вы знаете?
- Что такое такт квантования? Как такт квантования влияет на основные показатели качества?
- На что влияет коэф. передачи регулятора?
- Как в системе с ограничением могут возникнуть автоколебания? Как влияет такт квантования на амплитуду и частоту автоколебаний?
- Зависит ли временное и фазовое запаздывание от частоты входного сигнала.
- Чем объяснить зависимость переходного процесса от величины входного воздействия?

6. Список литературы

1. Техническое описание стенда (смотри лабораторную работу №1).

2. Конспект лекций по линейной теории автоматического управления.

3. Конспект лекций по нелинейной теории автоматического управления.

4. Следящие приводы: В 3т. 2-е изд., доп. и перераб. /Под ред. Б. К. Чемоданова. Т. I: Теория и проектирование следящих приводов /Е. С. Блейз, А. В. Зимин, Е. С. Иванов и др. – М.: Изд-во МГТУ им. Н. Э. Баумана, 1999. – 904 с.

Лабораторная работа № 8. Цифровая система управления приборного электрического следящего привода постоянного тока. Синтез цифрового фильтра.

Цель работы:

1. Синтез Z-передаточной функции корректирующего устройства с помощью пакета Matlab.

2. Программирование Z-передаточной функции.

3. Получение основных характеристик скорректированного ЦЭСП.

1. Техническое описание

Классический способом синтеза является синтез методом частотных характеристик. Однако в системах с цифровым вычислителем непосредственно использовать этот простой метод не удается. Один из путей решения этой проблемы – использование Z-преобразования, далее выполнение билинейного преобразования и переход к псевдочастотным характеристикам, после чего получение передаточной функции фильтра и обратный переход к Z-передаточным функциям. Безусловно, этот путь является правильным, но, в то же время, достаточно трудоемким. Однако, если принять некоторые допущения, то можно значительно упростить эту процедуру.

На рисунке 1. представлена структурная схема привода с использованием аналогового фильтра. Передаточную функцию фильтра получим, используя исходную и желаемую ЛАЧХ разомкнутой системы.

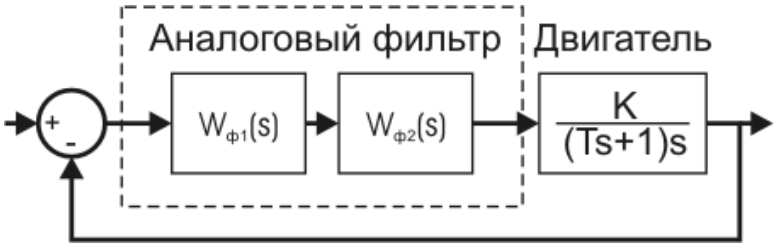


Рисунок 1. Упрощенная модель привода с аналоговым фильтром.

Исходная ЛАЧХ – это ЛАЧХ двигателя, который представлен апериодическим звеном и интегратором. На рисунке 2 представлена ЛАЧХ исходной (сплошная) и желаемой (штриховая) системы, а также показаны частоты всех изломов (ω_1, ω_1 …). Передаточная функция фильтра получается путем вычитания исходной ЛАЧХ из желаемой.

После того как получена передаточная функция фильтра, производим Z-преобразование и получим Z-функцию фильтра. Однако надо помнить об одном ограничении: такт квантования в системе должен быть хотя бы в 2 раза меньше самой маленькой постоянной времени системы. На рисунке 3 представлена эквивалентная схема системы, но уже с цифровым фильтром.

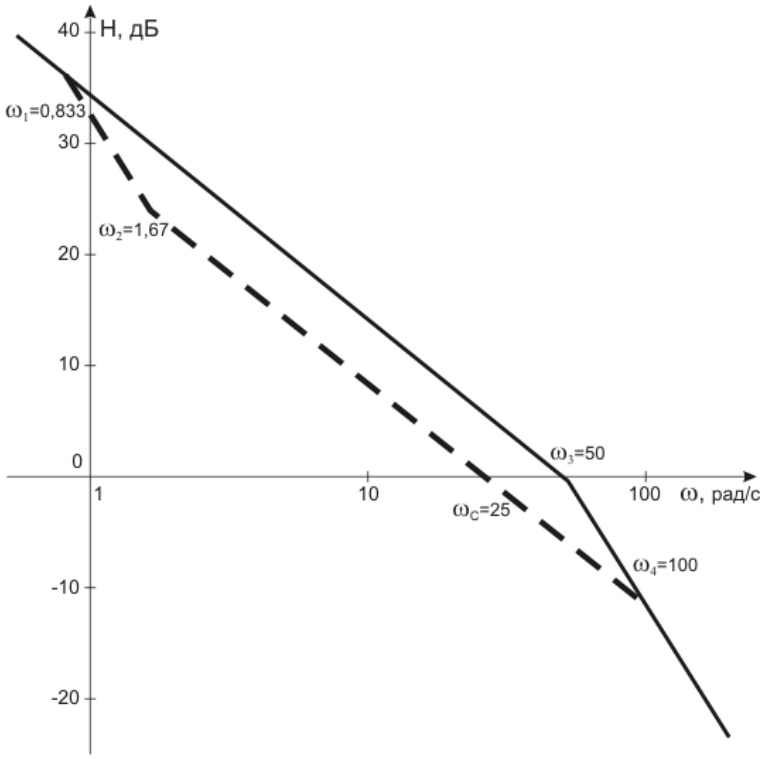


Рисунок 2. ЛАЧХ исходной (сплошная) и желаемой (штриховая) системы.

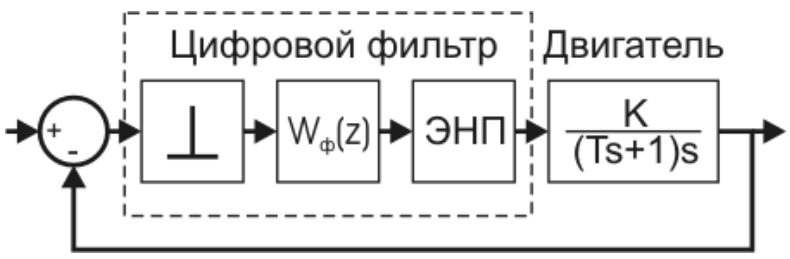


Рисунок 3. Упрощенная модель привода с цифровым фильтром.

Далее необходимо запрограммировать Z-функцию фильтра. Существует несколько способов выбора переменных состояния; будем использовать прямое, параллельное или последовательное программирование.

В результате получаем структурную схему, вводим переменные состояния и записываем в виде системы разностных уравнений. Далее необходимо выписать главную матрицу A, матрицу-столбец B, матрицу-строку C и матрицу D. Полученные матрицы вводятся в программу Scope.exe (смотри рисунок 4).

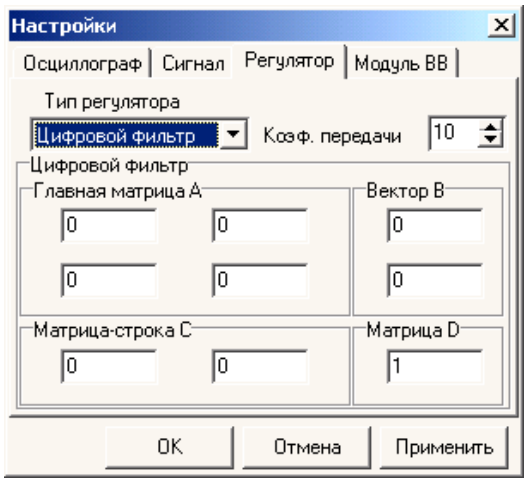


Рисунок 4. Окно ввода коэффициента передачи и матриц фильтра.

2. Приборы и оборудование

- Персональный компьютер с установленной системой Matlab;
- Приложение Scope.exe;
- Цифровой следящий привод.

3. Порядок выполнения работы.

- Передаточная функция двигателя имеет вид:

$$W\ddot{a}\hat{a}(s) = \frac{7.14}{(19.2 \cdot 10^{-3} s + 1)s}$$

По рисунку 2 необходимо определить величину коэффициента передачи по контуру. А затем получить коэффициент передачи фильтра K.

- После этого открываем модель Simulink и вводим полученный коэффициент передачи (смотри рисунок 5). Сохраняем переходный процесс в отчет.

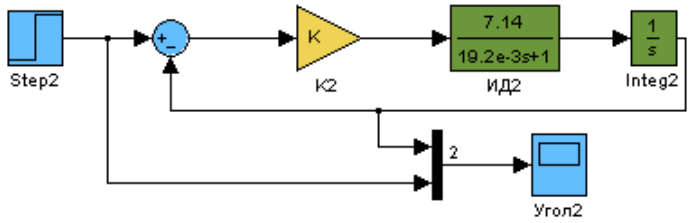


Рисунок 5. Упрощенная модель с П-регулятором.

- Необходимо получить передаточную функцию фильтра вычитанием исходной ЛАЧХ из желаемой (смотри рисунок 2). В результате должна получится передаточная функция вида:

$$W(s) = \frac{(T2s+1)(T3s+1)}{(T1s+1)(T4s+1)} \quad (1)$$

- Ввести полученный аналоговый фильтр в модель Simulink (смотри рисунок 6). Сохранить переходный процесс в отчет.

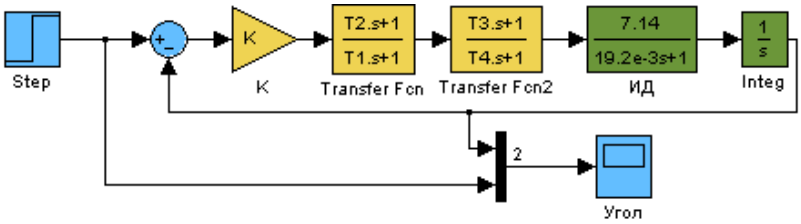


Рисунок 6. Модель с аналоговым фильтром.

- В командной строке Matlab создать переменную s – оператор дифференцирования. Для этого необходимо воспользоваться функцией tf (transfer function - передаточная функция): s= tf('s')
- Далее вводим передаточную функцию в виде: Ws= (T2*s+1)*(T1*s+1)/((T3*s+1)* (T4*s+1))
- где T1, T2 … T4 – конкретные числа из формулы (1).
- Далее производим Z-преобразование с помощью функции c2d (continues to discrete): Wz= c2d(Ws, T)

где Ws – введенная ранее переменная; T – такт квантования, с. Для совместимости со стендом примем T= 5 мс.

- Вводим полученный цифровой фильтр в модель Simulink (смотри рисунок 7). Сохраняем переходный процесс в отчет.

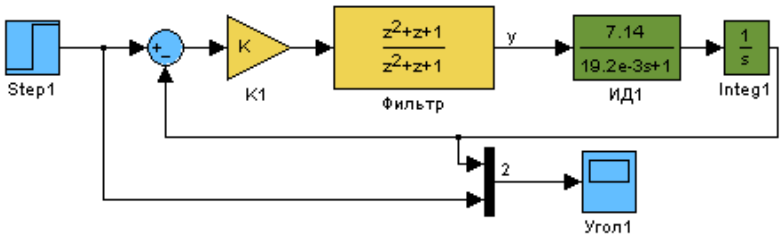


Рисунок 7. Модель с цифровым фильтром.

- После того как получена верная Z-передаточная функция, необходимо запрограммировать её одним из указанных преподавателем способов (прямое, параллельное или последовательное программирование). Записать в отчет полученную структурную схему.
- Записать систему разностных уравнений и выписать главную матрицу A, матрицу-столбец B, матрицу-строку C и матрицу D.
- Включить тумблер +27 В на панели блока «Усилитель, генератор, демодулятор».
- Вызвать окно «Настройки», выбрать закладку «Сигнал» и установить такт квантования 5 мс, амплитуду сигнала 50 мрад и 120 мрад. На закладке «Регулятор» выбрать тип регулятора «Цифровой фильтр», установить полученный коэф. передачи и ввести матрицы A, B, C и D. Зафиксировать изменения и запустить управление. Занести результаты в отчет.
- Вызвать окно «Настройки», выбрать закладку «Сигнал» и установить амплитуду сигнала 50 мрад и 120 мрад. На закладке «Регулятор» выбрать тип регулятора «П-регулятор». Зафиксировать изменения и запустить управление. Занести результаты в отчет.
- Выключить двигатель. Закрыть программу. Выключить блок питания привода.

4. Содержание отчета

- Название и номер работы, цель работы;
- ЛАЧХ исходной и скорректированной системы, а также ЛАЧХ фильтра.
- Коэффициент передачи фильтра, передаточная функция аналогового фильтра, Z-передаточная функция цифрового фильтра,
- Структурная схема Z-передаточной функции цифрового фильтра
- Система разностных уравнений и матрицы A, B, C, D.
- Все переходные процессы с показателями качества, указанные в порядке выполнения работы;
- Выводы.

5. Контрольные вопросы

- Как по ЛАЧХ определить коэффициент передачи по контуру?
- Какой наклон должна иметь ЛАЧХ в районе частоты среза?
- Какой способ программирования наиболее выгоден? Почему?
- Почему матрица B в данной системе является матрицей столбцом, а не прямоугольной матрицей? Что такое матрица D?

Список литературы

1. Руководство по системе Matlab.
2. Техническое описание стенда (смотри лабораторную работу №1).
3. Конспект лекций по линейной теории автоматического управления.
4. Конспект лекций по дискретной теории автоматического управления.