МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение
высшего профессионального образования -
«Оренбургский государственный университет»

Кафедра промышленной электроники и
информационно-измерительной техники

А.В.ХЛУДЕНЁВ

РАЗРАБОТКА И ОТЛАДКА УСТРОЙСТВ ДИСКРЕТНОГО УПРАВЛЕНИЯ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЕ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения высшего профессионального
образования -
«Оренбургский государственный университет»

Оренбург 2003

ББК 32.859 я 73
Х 60
УДК 621.38 (075.8)

Рецензент
доктор технических наук, доцент В.Н. Булатов

Х 60 **Хлуденёв А.В. Разработка и отладка устройств дискретного управления: Методические указания. - Оренбург: ОГУ, 2003. – 23 с.**

Методические указания предназначены для выполнения расчетно-графической работы для студентов, обучающихся по программам высшего профессионального образования по специальности 200400 "Промышленная электроника" при изучении дисциплины "Отладочные средства микропроцессорных систем"

ББК 32.859 я 73

1 Цели и задачи расчетно-графической работы

Цель выполнения работы - подготовка к самостоятельной профессиональной деятельности в области разработки и отладки микропроцессорных устройств (МПУ), основанной на системном использовании средств автоматизации.

В результате выполнения расчетно-графической работы студент должен:

- иметь представление о состоянии и перспективных направлениях развития методов и средств разработки и отладки микропроцессорной техники;
- знать методы программной реализации алгоритмов дискретного управления;
- уметь использовать методы и средства отладки микропроцессорной техники;
- иметь опыт разработки и отладки программного обеспечения МПУ.

2 Организация выполнения расчетно-графической работы

Над расчетно-графической работой студенты работают под руководством преподавателя в часы, отводимые расписанием, и самостоятельно в свободное от занятий время. Основная роль отводится самостоятельной работе студентов. Решение задач, требующих использования средств автоматизации разработки и отладки программного обеспечения, выполняется в компьютерном классе. Возникающие в процессе работы над работой вопросы студенты могут решать в часы консультаций с преподавателем. Планируемое время выполнения работы составляет 8 часов.

Отчет о работе студент сдает на проверку преподавателю не менее чем за 3 дня до защиты. Без предварительной проверки отчетов работы к защите не допускаются. Преподаватель в течение 1-2 дней проверяет отчет и возвращает его студенту с замечаниями, которые студент должен устранить, или подписанным, если работа допущена к защите.

При защите работы студент должен продемонстрировать работоспособность разработанной программы МПУ, используя средства автономной и комплексной отладки, и ответить на вопросы. При оценке работы принимается во внимание:

- знания, умения и навыки студента в области автоматизации разработки и отладки программного обеспечения МПУ;
- качество полученных проектных решений;
- обоснование принятия неформальных решений, касающихся выбора различных вариантов реализации, способов тестирования;
- степень самостоятельности при выполнении работы и освоения использованных инструментальных средств.

3 Задание на расчетно-графическую работу

Студент должен разработать и отладить программные средства МПУ, предназначенного для дискретного управления техническим объектом. Алгоритм управления задан в виде кодированной граф-схемы в соответствии с вариантом задания. Основные проектные операции и процедуры должны быть выполнены с использованием средств автоматизации разработки и отладки программного обеспечения МПУ. Корректность полученных проектных решений и выполнение требований задания должны быть подтверждены результатами тестирования.

Отчет по работе должен содержать:

- обоснование принятых решений по разработке и отладке программных средств МПУ;
- исходный текст программы;
- протокол трансляции с языка ассемблера;
- карту загрузки;
- результаты автономной и комплексной отладки;
- выводы.

4 Содержание расчетно-графической работы

Одним из направлений применения микропроцессоров (МП) является замена аппаратных (схемных) средств реализации логических функций и автоматов с памятью. При этом использование МП изменяет характер проектирования: из области разработки аппаратных средств (жесткой логики) усилия проектировщика переносятся в область программирования. Применение МП увеличивает гибкость систем, поскольку переход к решению новых задач не требует переделки аппаратуры и обеспечивается загрузкой в память микропроцессорной системы новых программ.

Объектом проектирования в работе является устройство дискретного управления. Это цифровое устройство, вырабатывающее последовательность наборов выходных двоичных сигналов. Последовательность формируемых двоичных наборов, как правило, зависит от последовательности поступления наборов входных двоичных сигналов и определяется алгоритмом функционирования. Устройства данного класса могут быть реализованы на жесткой или программируемой логике (как микропрограммные автоматы) или на универсальных программируемых автоматах – МП или микроконтроллерах (МК).

Алгоритм функционирования устройств данного класса обычно задается граф – схемой алгоритма (ГСА). В общем случае ГСА содержат одну начальную, одну конечную и произвольное количество операторных и условных вершин. Выходные наборы в ГСА представляются операторными вершинами. Если в операторной вершине указан символ Y_i, то соответствующий разряд выходного набора должен принять значение

логической единицы, в противном случае - логического нуля. Условные вершины представляют операции проверки значений входных сигналов. В зависимости от результатов проверки формируется та или иная последовательность выходных наборов. Если в условной вершине указан символ X_i, то проверяется значение соответствующего разряда входного набора.

Предполагается, что аппаратные средства (АС) МПУ выполнены на МП типа КР580ВМ80 (КР1821ВМ85), и в качестве их прототипа используется учебно-отладочное устройство типа УМК с модулем расширения ТЭЗ-1М. Входные сигналы подаются на выводы порта *B* параллельного интерфейса КР580ВВ55 (адрес порта – 01h). Выходные сигналы формируются на выводах порта *A* (адрес порта – 00h).

Требуется разработать и отладить программные средства (ПС) МПУ. Программа должна быть загружена в интегральную схему постоянного замонинающего устройства (ПЗУ) типа К573РФ2 (К573РФ5), которой выделен сегмент адресного пространства 8000h – 87FFh.

Задание должно быть выполнено с использованием следующих средств:

- инструментального вычислительного комплекса (ПЭВМ) с базовым программным обеспечением;
- эмулятора ПЗУ;
- программатора ПЗУ с управляющей программой;
- кросс-системы AVOSET-85.

Правильность выполнения задания контролируется выполнением тестирования в среде симулятора-отладчика и на прототипе МПУ. В ходе тестирования устанавливается соответствие работы МПУ заданному алгоритму функционирования. В случаях, когда в результате тестирования выявляется несоответствие заданному алгоритму функционирования, необходимо локализовать и устранить допущенные ошибки или неточности.

5 Методические указания

5.1 Способы программной реализации алгоритма дискретного управления

На порт ввода МПУ поступает набор входных сигналов $\boldsymbol{X}=<x_1,x_2,\ldots,x_n>$. Командой ввода $\boldsymbol{X}$ заносится в память (регистры) МП. По хранимой в ПЗУ программе МП вычисляет требуемые значения набора выходных сигналов $\boldsymbol{Y}=<y_1,y_2,\ldots,y_m>$ и командой вывода заносит его в порт вывода.

Рассмотрим известные методы программной реализации алгоритма дискретного управления /1, 2/. Они требуют, чтобы алгоритм функционирования был задан в автоматной форме.

Устройство дискретного управления можно представить моделью в форме автомата Мили или автомата Мура. Чтобы получить автоматную форму, необходимо на ГСА отметить состояния автомата. Состояниям автомата Мура сопоставляются (отмечаются на ГСА) все операторные вершины. Начальная и

конечная вершины отмечаются одним состоянием. Они ставятся в соответствие исходному состоянию автомата, в котором он оказывается при инициализации. Состояниям автомата Мили сопоставляются подмножества входов вершин, непосредственно следующие за операторными вершинами. Выход начальной и вход конечной вершины отмечаются одним состоянием. Они также ставятся в соответствие исходному состоянию автомата. Если хотя бы один из входов конечной вершины соединен с выходом операторной вершины, то для обеспечения эквивалентности автомата Мили автомату Мура необходимо перед конечной вершиной ввести пустую операторную вершину.

Для реализации автомата необходимо выполнить кодирование его внутренних состояний. Для рассматриваемых микропроцессорных вариантов реализации кодирование может быть произвольным. Число разрядов k кодов внутренних состояний $\boldsymbol{Q}$ определяется числом состояний автомата N_s

$$k = ent(\log_2 N_s),$$

где *ent* – операция округления до ближайшего большего целого числа.

Табличный метод реализует представление алгоритма автомата в форме таблицы переходов-выходов. Так как значения функций переходов и выходов автомата зависят от входных сигналов $\boldsymbol{X}$ и кодов внутренних состояний $\boldsymbol{Q}$, то каждому сочетанию входных наборов и внутренних состояний необходимо поставить в соответствие следующее состояние и выходной набор. Общее число таких комбинаций составит $2^n \cdot N_s$, а объем памяти для хранения кодов состояний и выходных наборов составит $2^n \cdot N_s \cdot (m+k)$.

Программная реализация этого метода сводится к формированию в зависимости от входного слова адреса, по которому в памяти находится соответствующее выходное слово. Здесь под входным (выходным) словом понимается результат соединения (конкатенации) входного (выходного) набора и кода текущего состояния автомата. Наиболее просто это можно организовать посредством относительной адресации. Для этого используют регистр, выполняющий функцию индексного регистра. В этот регистр предварительно записывают начальный адрес таблицы соответствия, а в качестве смещения используют числовой эквивалент входного слова.

При неполном кодировании состояний автомата (если $2^k > N_s$) часть ячеек памяти таблицы соответствия не используется. Память используется нерационально. Исключение этих ячеек приведет к нарушению правильности адресации при рассмотренном способе формирования исполнительного адреса.

Отмеченного недостатка лишен метод логического ветвления. Он состоит в реализации переходов автомата из одного состояния в другое по условию равенства числового эквивалента входного слова некоторой константе, поставленной в соответствие определенному выходному слову. Недостатком метода логического ветвления является более низкое быстродействие по сравнению с табличным методом, а также зависимость периода опроса входных сигналов от выполняемого перехода.

В большинстве случаев период опроса входных сигналов задается или определяется особенностями задачи управления. Если время выполнения цикла программной реализации алгоритма получается меньше требуемого значения периода, то необходимо использовать программную реализацию дополнительной задержки. Если время выполнения цикла программной реализации алгоритма получается больше требуемого значения периода, то необходимо использовать более быстродействующий МП или МК.

5.2 Пример реализации табличным методом

На рисунке 1 приведена ГСА дискретного управления объектом с отметкой внутренних состояний автомата Мили. Внутренние состояния обозначены метками *A0 – A3*.

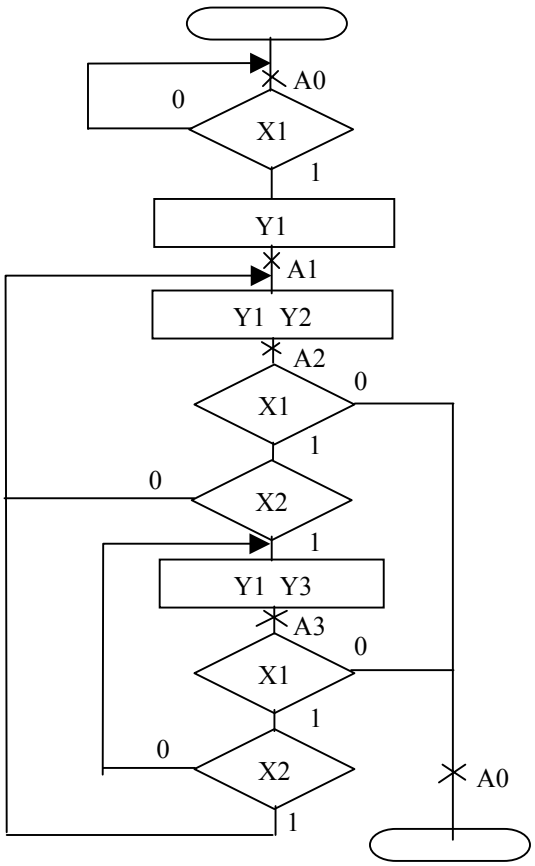

Рисунок 1

Алгоритм автомата представим в виде функций переходов и выходов (таблица 1). Состояния автомата Мили после выполнения перехода и текущие значения выходов зависят от исходного состояния автомата A_i и значений входных переменных x_2 x_1. Их значения приведены соответственно над и под наклонной чертой. Значения выходных наборов приведены в последовательности y_3 y_2 y_1.

Таблица 1

A_i	$x_2\,x_1$			
	00	01	10	11
A0	A0/0 0 0	A1/0 0 1	A0/0 0 0	A1/0 0 1
A1	A2/0 1 1	A2/0 1 1	A2/0 1 1	A2/0 1 1
A2	A0/0 0 0	A2/0 1 1	A0/0 0 0	A3/1 0 1
A3	A0/0 0 0	A3/1 0 1	A0/0 0 0	A2/0 1 1

Число внутренних состояний автомата $N_s = 4$, тогда число разрядов кода

$k = ent(\log_2 4) = 2.$

Для кодирования состояний введем двоичные переменные $q_1\ q_0$. Общее число разрядов во входных наборах и кодах внутренних состояний

$N_{in} = n + k = 2 + 2 = 4 < 8$,

поэтому эту информацию можно представить однобайтовым входным словом

$0\ 0\ 0\ 0\ x_2\ x_1\ q_1\ q_0$.

Число разрядов в кодах внутренних состояний и выходных наборах

$N_{out} = k + m = 2 + 3 = 5 < 8$,

поэтому эту информацию также можно представить однобайтовым выходным словом

$0\ 0\ 0\ y_3\ y_2\ y_1\ q_1\ q_0$.

Выполним кодирование состояний автомата (таблица 2) и составим таблицу соответствия входных и выходных слов (таблица 3).

Таблица 2

Состояние	$q_1\ q_0$
A0	00
A1	01
A2	10
A3	11

Исходный текст программы на языке ассемблера, в которой реализован табличный метод, приведен в приложении А. Текст программы содержит описания констант, портов ввода-вывода и алгоритма выполнения программы.

Элементы программы размещаются в трех сегментах, определенных по умолчанию:

- IOSPACE сегмент пространства портов ввода-вывода;
- CODE – сегмент кода программы;
- DATA – сегмент данных программы, содержащий таблицу соответствия.

Для хранения кода внутреннего состояния используются младшие разряды регистра *B*. Начальный адрес таблицы соответствия хранится в регистровой паре *HL*. Программная задержка реализована на вычитающем счетчике, выполненном на регистровой паре *DE*. Константа *DEL* определяет величину задержки, необходимую для обеспечения периода опроса входных сигналов 12 мс.

Таблица 3

Входное слово	Выходное слово	Входное слово	Выходное слово
00000000	00000000	00001000	00000000
00000001	00001110	00001001	00001110
00000010	00000000	00001010	00000000
00000011	00000000	00001011	00000000
00000100	00000101	00001100	00000101
00000101	00001110	00001101	00001110
00000110	00001110	00001110	00010111
00000111	00010111	00001111	00001110

5.3 Пример реализации методом логического ветвления

В качестве исходных данных используем полученную таблицу соответствия входных и выходных слов (таблица 3). Поставим каждому выходному слову множество соответствующих ему входных слов. Причем входные слова удобнее представить их десятичными эквивалентами. Результат представлен в таблице 4.

Таблица 4

Выходное слово	Эквиваленты входных слов
00000000 (0)	0,2,3,8,10,11
00000101 (5)	4,12
00001110 (14)	1,5,6,9,13,15
00010111 (23)	7,14

Исходный текст программы на языке ассемблера целевого МП, реализующей метод логического ветвления, приведен в приложении Б. Требуемое значение выходного слова заносится в регистр *B* при переходе на метки *L0, L5, L14, L23*. Полученное значение используется для формирования значения выходного набора и нового внутреннего состояния автомата, которое сохраняется в регистре *B*.

5.4 Отладка программных средств

Автономная отладка разработанного варианта ПС для МПУ выполняется средствами кросс-системы AVOSET-85.

Для выполнения работы необходимо ввести исходный текст программы на языке ассемблера целевого МП в любом текстовом редакторе и сохранить в файле с расширением .asm. Затем выполняется трансляция программы с помощью Avmac85.exe. Программа Avmac85.exe запускается на выполнение с командной строки. Формат запуска:

Avmac85 FileName.asm

где FileName - имя файла программы на языке ассемблера.

При отсутствии синтаксических ошибок результат трансляции формируется в виде объектного модуля программы (файл с расширением .obj). Отчет о трансляции записывается в файл протокола с расширением .prn. Протокол трансляции программы Tabl приведен в приложении В. Символы, стоящие в протоколе после значений адресов, указывают, что трансляция выполнена в перемещаемые сегменты:

- CODE – символ «’»;
- DATA – символ «”»;
- IOSPACE – символ «&».

Абсолютные значения адресов формируются при редактировании связей программой Avlink.exe. Следует учитывать, что отлаженная программа должна располагаться в заданной области адресного пространства МП (8000h – 87FFh), но при этом занимать физические адреса ПЗУ программ 0000h – 07FFh. Необходимое смещение обеспечивается дешифратором адреса МПУ. Поэтому привязку программы к требуемой области адресного пространства МП целесообразно выполнить на этапе редактирования связей. Причем потребуется получить два варианта загрузочного модуля - один для отладки на программно-логической модели МПУ, а второй – на прототипе МПУ. Формат запуска:

Avlink [options] FileName=FileName.obj

где FileName - имя файла-результата;

FileName.obj - имя файла объектного модуля);

options - опции, которые можно вставить в любое место командной строки. В качестве опций следует указать;

-SY, чтобы создать файл с таблицей символов;

-ST(class, addr), чтобы установить требуемый стартовый адрес программы для отладки на программно-логической модели МПУ;

-RA(seg, addr), чтобы установить стартовый адрес программы для выполнения на прототипе МПУ;

-H, чтобы получить справку по Avlink. В данном случае формат запуска: Avlink –ST(M,8000h) – SY FileName=FileName.obj

Результатом работы программы будет файл загрузочного модуля FileName.hex (отключается опцией -NO), FileName.map (карта загрузки, отключается опцией -NM), FileName.sym (таблица символов, если была указана опция -SY). Загрузочный модуль будет расположен, начиная с адреса 8000h.

Отладка программы выполняется путем проверки ее работы для различных наборов исходных данных и сопоставления полученных результатов с ожидаемыми. По результатам этого сопоставления делают заключение о правильности работы программы или наличии ошибок, которые необходимо локализовать и исправить в исходном тексте программы. При этом процесс отладки носит итерационный характер. Как правило, сначала отладку проводят на программно-логической модели МПУ в среде симулятора-отладчика.

Объект отладки относится к цифровым устройствам последовательностного типа, у которых устойчивые значения выходных сигналов зависят от комбинации значений входных сигналов и от внутреннего состояния. Если автомат имеет n входов и N_s внутренних состояний, то для полного тестирования необходимо выполнить $2^n \cdot N_s$ вариантов моделирования.

Перед выполнением тестирования следует составить его план в виде упорядоченной последовательности входных наборов, которая позволяет выявить полное соответствие функционирования МПУ заданному алгоритму.

Рассчитанные значения выходных наборов необходимо сравнить с соответствующими значениями для заданного алгоритма функционирования и сделать вывод об их совпадении или о существовании отличий.

Рассмотрим формирование теста для программной реализации рассмотренного ранее алгоритма. Для автомата Мили алгоритмом предусмотрено выполнение переходов:

- A0 → A0; A0 → A1;
- A1 → A2;
- A2 → A0; A2 → A2; A2 → A3;
- A3 → A3; A3 → A2; A3 → A0.

Сформируем цепочку из этих переходов, чтобы в ней были все переходы, а общее число переходов было минимально, например:

A0 → A0 → A1 → A2 → A0 → A1 → A2 → A2 → A3 → A2 → A3 → A3 → A0.

Для данной последовательности состояний сформируем тест, задавая последовательность соответствующих значений входных сигналов, необходимых для выполнения каждого перехода

```
X1: 01X01X111110
X2: XXXXXX01110X
```

Символ ‘X’ использован для обозначения произвольного значения сигналов. Для выполнения полного тестирования необходимо исследовать поведение МПУ для различных комбинаций значений этих сигналов (0 или 1). Частичное тестирования достаточно выполнить для одной комбинации, например,

```
X1: 011011111110
X2: 000000011100
```

После запуска симулятора-отладчика AvSim85.exe в режиме A необходимо загрузить отлаживаемую программу командой "Load\Avocet > FileName". Переход к первой команде выполняется путем загрузки в счетчик команд PC значения начального адреса программы 8000h. Так как отлаживаемая программа является циклической, и в каждом цикле выполняется ввод входного набора и вывод выходного, то тестирование целесообразно выполнить следующим образом.

Командой Set\Sticky\A-R/W point установить постоянную точку останова на команде JMP LOOP. Затем для каждого тестового входного набора:

- устанавить соответствующие значения разрядов порта 01h;
- выполнить прогон программы до точки останова (клавиша F1);
- по состоянию разрядов порта 00h определить значения выходного набора и по содержимому регистра B – следующее состояние автомата;
- сравнить полученные значения с заданными алгоритмом;
- если значения совпали, перейти к метке LOOP (клавиша F10) и продолжить тестирование для следующего входного набора, если нет, то определить и устранить причину ошибки.

При большом числе входных наборов тестирование программы может занять достаточно много времени. Этот процесс можно значительно ускорить, если использовать возможность работы симулятора-отладчика с потоками входных и выходных данных МПУ, представленных в виде файлов. Чтобы использовать эту возможность, необходимо подготовить двоичный файл, содержащий последовательность наборов, поступающих на входной порт МПУ. Значения этих наборов необходимо определить с учетом положения входных сигналов в разрядной сетке порта.

В рассматриваемом примере сигналы $x_2\, x_1$ должны приходить соответственно на 3-й и 2-й разряды порта, поэтому значения входных наборов можно получить, умножая на 4 десятичные эквиваленты комбинаций сигналов $x_2\, x_1$.

Получим последовательность значений: 0 4 4 0 4 4 4 12 12 12 4 0.

Эту последовательность чисел можно набрать в текстовом редакторе и сохранить в текстовом файле с расширением .ad (например, In.ad), двоичное представление данных получается в результате обработки программой IOTran.exe. Формат запуска:

IOTran.exe db In

где db – параметр, определяющий направление преобразования данных (из десятичной формы в двоичную);

In – имя входного файла с расширением .ad. Результат преобразования будет представлен в файле с расширением .ab (In.ab).

Созданный файл необходимо открыть из среды симулятора-отладчика командой IO\Open\IO Input File, указав его имя, и отрицательно ответить на вопрос Replay Input File at EOF (циклически повторять чтение при достижении конца файла). Далее необходимо проигнорировать приглашение IO Output File, нажав клавишу Enter, и задать условие считывания одного элемента входной последовательности (IO Trigger):

- через заданное число машинных тактов (Cycles);
- при выполнении команд чтения/записи (Opcode access). В данном случае больше подходит второй вариант. После его выбора необходимо через запятую задать значения параметров Address, Dir: IN/OUT (адрес порта, направление: ввод/вывод). Например: I:1,IN (порт 01h, ввод).

Аналогично можно открыть выходной файл, например OUT.AB, командой IO\Open\IO Output File (проигнорировав приглашение IO Input File) и задать условие записи одного элемента выходной последовательности I:0,OUT (порт 00h, вывод).

Для получения результата необходимо запустить программу на выполнение (клавиша F1) без установки точек останова. Прерывание выполнения программы происходит при считывании всех элементов входного файла, о чем свидетельствует сообщение Trap: In.ab Empty. После этого входной и выходной файлы необходимо закрыть командой IO\Close. Для выполнения анализа выходной последовательности выходной двоичный файл можно преобразовать в десятичный формат программой IOTran.exe. Формат запуска:

IOTran.exe bd OUT

Результат преобразования будет сохранен в файле OUT.AD. Опыт эксплуатации симулятора-отладчика AVSim85 показывает, что первый элемент, выходной последовательности может записываться до выполнения условия записи, в этом случае его не следует принимать во внимание.

Если для всей последовательности наборов результаты совпадают с заданными алгоритмом, то можно считать, что программа не содержит грубых ошибок и работает правильно.

Средства симулятора-отладчика также позволяют приближенно оценить длительность периода считывания входных наборов. Сделать это можно путем определения количества машинных тактов между двумя последовательными операциями чтения порта ввода. Для этого необходимо:

- установить счетчик команд PC на команду перехода JMP LOOP;

- инициализировать счетчик машинных тактов Cycles командой Set\cYcles;

- установить на команде IN PB динамическую точку останова клавишей F3 (предварительно перемещая курсор точки останова клавишами F2, F4);

- запустить программу на выполнение клавишей F1 до достижения точки останова;

- выполнить команду JMP LOOP в пошаговом режиме (клавиша F10);

- считать содержимое счетчика машинных тактов Cycles.

Значение периода опроса входного порта можно определить, зная длительность машинного такта T_m,

$$T = Cycles \cdot T_m .$$

Следует иметь в виду, что:

- модуль счетчика тактов равен 65536, поэтому при больших значениях задержки он может переполняться и неоднократно (коррекция показаний – число переполнений умножить на 65536);

- МП типов КР580ВМ80 и КР1821ВМ85 имеют практически одну систему команд, но время выполнения одних и тех же команд (выраженное в машинных тактах) у них может отличаться. Поэтому для МП типа КР580ВМ80 значение периода Т будет определено приближенно. Более точно это значение можно определить в процессе комплексной отладки, причем с учетом действительного значения тактовой частоты МП.

Комплексную отладку выполняют на прототипе МПУ, при этом программа должна быть загружена в ПЗУ. Следует учитывать, что даже отлаженная автономно программа может содержать ошибки. Поэтому эту задачу часто приходится выполнять неоднократно, стирая и перепрограммируя ПЗУ. Проблема решается за счет использования специальных средств комплексной отладки: схемных эмуляторов и эмуляторов ПЗУ. В этом случае в ПЗУ записывается уже отлаженная программа.

Для отладки на прототипе МПУ необходимо подготовить другой варианет загрузочного модуля при запуске редактора связей с параметрами

Avlink –RA(CODE,8000h) – SY FileName=FileName.obj

Для записи в ПЗУ или для загрузки в его эмулятор необходимо получить двоичный код загрузочного модуля программы. С этой целью можно воспользоваться программой Hexform.exe. Формат ее запуска с командной строки

Hexform FileName=FileName.hex

где FileName - имя файла-результата (присваивается расширение .bin);
FileName.hex - имя файла загрузочного модуля в hex-формате.

При работе с эмулятором ПЗУ в соответствующую панельку на прототипе устанавливают эмуляционный разъем, включают питание прототипа и запускают на инструментальном компьютере управляющую программу Emul.exe. Для загрузки программы необходимо выбрать тип эмулируемого ПЗУ, задать начальный адрес загрузки (в адресном пространстве ПЗУ!) и нажать кнопку «Загрузка ОЗУ».

Для программирования ПЗУ, программатор следует подключить к параллельному порту компьютера и запустить управляющую программу pro_gg.exe. При работе в среде программы сначала необходимо выбрать тип ИС ПЗУ и соответствующее значение напряжения программирования. Затем следует установить в панельку программатора ИС ПЗУ выбранного типа и включить программатор.

Следует убедиться в пригодности ИС ПЗУ к программированию (режим – «Контроль ПЗУ»). При необходимости можно выполнить чтение ПЗУ в выбранном секторе и просмотреть содержимое сектора в окне программы. Далее выполняется загрузка кода программы в ОЗУ программатора (команда LOAD), для этого необходимо указать имя двоичного файла с загрузочным модулем программы МПУ. При необходимости можно просмотреть содержимое сектора ОЗУ в окне программы. После этого выполняется непосредственно программирование ПЗУ. Результат программирования проверяется путем сравнения содержимого ПЗУ и содержимого ОЗУ программатора.

ПЗУ с программой МПУ извлекается из панельки выключенного программатора и устанавливается в соответствующую панельку ПЗУ на прототипе МПУ.

Тестирование на прототипе МПУ целесообразно выполнять в режиме прогона с точками останова или в пошаговом режиме. Запуск программы на прототипе МПУ выполняется последовательным нажатием клавиш

«СТ» адрес1 «_ » адрес2 «ВП»,

где адрес1 – стартовый адрес программы;

адрес2 – адрес точки останова. Точку останова в рассматриваемом примере можно установить на команде JMP LOOP.

Значения входных наборов задают с помощью микропереключателей, подключенных ко входам порта В, а значения выходных наборов по индикаторам, подключенным к выходам порта А. Программу МПУ можно считать отлаженной на прототипе, если для всей последовательности входных наборов значения выходных наборов совпадают с заданными алгоритмом.

Длительность периода считывания входных наборов можно определить при выполнении программы в реальном времени с помощью измерителя временных интервалов по изменениям выходных сигналов, например, при $x_2 = 1$, $x_1 = 1$. Вместо измерителя временных интервалов можно использовать осциллограф, но погрешность измерения составит около 5 %.

5.5 Контрольные вопросы для защиты

5.5.1 Сравните программную реализацию алгоритмов дискретного управления с реализацией на жесткой и программируемой логике. Перечислите преимущества и недостатки.

5.5.2 Перечислите методы программной реализации алгоритмов дискретного управления. Кратко поясните их содержание.

5.5.3 Дайте сравнительную оценку этим методам.

5.5.4 Перечислите использованные средства разработки и отладки ПС МПУ.

5.5.5 Перечислите этапы разработки ПС МПУ.

5.5.6 Какие приемы используются для выполняется автономной и комплексной отладки ПС МПУ?

5.5.7 Как сформировать тест для отладки МПУ, реализующего алгоритм дискретного управления?

Список использованных источников

1. Герасимов И.В. Построение цифровых устройств автоматики и вычислительной техники на современной элементной базе. – Л., ЛЭТИ, 1982. - 49 с.

2. Угрюмов Е.П. Проектирование элементов и узлов ЭВМ. - М.: Высш. шк., 1987. - 318 с.

3.

Приложение А
(обязательное)
Исходный текст программы Tabl.asm

```
DEL   EQU   1000      ; значение задержки

      SEG IOSPACE, START=00h
PA:   DS    1         ; порты ИС КР580ВВ55
PB:   DS    1
PC:   DS    1
PS:   DS    1

      SEG CODE, START=8000h
      MVI   A,8Bh     ; инициализация порта
      OUT   PS
      MVI   B,0       ; начальное состояние
LOOP: IN    PB        ; прием входного набора
      ANI   00001100b ; выделение x2x1
      ORA   B         ; формирование входного слова и
      MOV   C,A       ; запись его в C
      LXI   H,TAB     ; загрузка начальн. адреса таблицы
      MVI   B,0
      DAD   B         ; формирование адреса строки
      MOV   B,M       ; выборка выходного слова
      MOV   A,B
      ANI   00011100b ; выделение выходного набора
      OUT   PA        ; вывод выходного набора
      MOV   A,B
      ANI   00000011b ; выдел. кода сост.
      MOV   B,A       ; и запись его в B
      LXI   D,DEL     ; цикл задержки
DLY:  DCX   D
      MOV   A,D
      ORA   E
      JNZ   DLY
      JMP   LOOP

      SEG DATA        ; таблица соответствия
TAB   DB    00h, 0Eh, 00h, 00h, 05h, 0Eh, 0Eh, 17h
      DB    00h, 0Eh, 00h, 00h, 05h, 0Eh, 17h, 0Eh
     END
```

Приложение Б
(обязательное)
Исходный текст программы LogBranch.asm

```
DEL    EQU   1000       ; значение задержки

       SEG IOSPACE, START=00h
PA:    DS    1          ; порты ИС КР580ВВ55
PB:    DS    1
PC:    DS    1
PS:    DS    1

       SEG CODE, START=8000h
MVI    A,    8Bh        ; инициализация порта
       OUT   PS
       MVI   B,0        ; начальное состояние
LOOP:  IN    PB         ; прием входного набора
       ANI   00001100b  ; выделение x2 x1
       ORA   B          ; формир. входного слова и
       MOV   C,A        ; запись его в C
       CPI   0
       JZ    L0
       CPI   2
       JZ    L0
       CPI   3
       JZ    L0
       CPI   8
       JZ    L0
       CPI   10
       JZ    L0
       CPI   11
       JZ    L0
       CPI   4
       JZ    L5
       CPI   12
       JZ    L5
       CPI   1
       JZ    L14
       CPI   5
       JZ    L14
       CPI   6
       JZ    L14
       CPI   9
       JZ    L14
       CPI   13
       JZ    L14
       CPI   15
```

```
        JZ   L14
        CPI  7
        JZ   L23
        CPI  14
        JZ   L23
L0:     MVI  B,0
        JMP  OUTP
L5:     MVI  B,5
        JMP  OUTP
L14:    MVI  B,14
       JMP   OUTP
L23:    MVI  B,23
OUTP:   MOV  A,B
        ANI  00011100b   ; выделение выходного набора
        OUT  PA          ; вывод выходного набора
        MOV  A,B
        ANI  00000011b   ; выделение кода состояния
        MOV  B,A         ; и запись его в B
        LXI  D,DEL       ; цикл задержки
DLY:    DCX  D
        MOV  A,D
        ORA  E
        JNZ  DLY
        JMP  LOOP
        END
```

Приложение В
(обязательное)
Протокол трансляции программы Tabl.asm

```
=03E8                1  DEL   EQU   1000
                     2
                     3        SEG   IOSPACE, START=00h
0000& (0001)         4  PA:   DS    1
0001& (0001)         5  PB:   DS    1
0002& (0001)         6  PC:   DS    1
0003& (0001)         7  PS:   DS    1
                     8
                     9        SEG   CODE, START=8000h
0000' 3E 8B         10        MVI   A,8Bh
0002' D3 03&        11        OUT   PS
0004' 06 00         12        MVI   B,0
0006' DB 01&        13  LOOP: IN    PB
0008' E6 0C         14        ANI   00001100b
000A' B0            15        ORA   B
000B' 4F            16        MOV   C,A
000C' 21 0000"      17        LXI   H,TAB
000F' 06 00         18        MVI   B,0
0011' 09            19        DAD   B
0012' 46            20        MOV   B,M
0013' 78            21        MOV   A,B
0014' E6 1C         22        ANI   00011100b
0016' D3 00&        23        OUT   PA
0018' 78            24        MOV   A,B
0019' E6 03         25        ANI   00000011b
001B' 47            26        MOV   B,A
001C' 11 03E8       27        LXI   D,DEL
001F' 1B            28  DLY:  DCX   D
0020' 7A            29        MOV   A,D
0021' B3            30        ORA   E
0022' C2 001F'      31        JNZ   DLY
0025' C3 0006'      32        JMP   LOOP
                    33
                    34        SEG DATA
0000" 00 0E 00      35  TAB   DB 00h,0Eh,00h,00h,05h,0Eh,0Eh
                              ,17h
0003" 00 05 0E
0006" 0E 17
0008" 00 0E 00      36        DB 00h,0Eh,00h,00h,05h,0Eh,17h
                              ,0Eh
000B" 00 05 0E
000E" 17 0E         37        END
```

Лицензия N ЛР020716 от 02.02.93
Подписано в печать .....
Формат ..... Бумага .....
Усл. печ. л. 2,0 Тираж .... Заказ .....

Отпечатано в Оренбургском государственном университете
460352, г. Оренбург, ГСП, пр. Победы, 13,
Издательство ОГУ