УДК 681.5 DOI: 10.17587/mau.19.232-242

Н. Г. Чикуров, канд. техн. наук, доц.,
Уфимский государственный авиационный технический университет

Синтез микропрограммных дискретно-логических систем управления

Разработана методика построения нового вида дискретно-логических микропрограммных систем управления. Дан пример построения микропрограммной системы управления в инструментальной среде программирования ISaGRAF.

Ключевые слова: *управление, дискретно-логические системы, электроавтоматика, металлорежущие станки*

Микропрограммный автомат Уилкса — Стринджера

Микропрограммирование обычно связывают с именем М. Уилкса (*Maurice Vincent Wilkes*), предложившего совместно с Дж. Стринджером (*John Bentley Stringer*) в 1951 г. микропрограммное управление в качестве упорядоченного и гибкого средства для управления вычислительными машинами [2]. Программа, задаваемая вычислительной машине, представляет собой последовательность команд. Микропрограммное управление означает, что машина с помощью заранее составленной внутренней микропрограммы интерпретирует каждую команду и затем исполняет ее в виде последовательности микроопераций.

Техническая база вычислительной техники долго сдерживала практическое применение микропрограммных автоматов, но в начале 60-х годов интерес к микропрограммному управлению резко возрос, и большинство машин стали микропрограммными.

Исходная схема, предложенная М. Уилксом и Дж. Стринджером, состоит из управляющей матрицы $\tilde{N}$, матрицы S, задающей последовательность действий, пирамидального дешифратора, на который подаются импульсы синхронизации, и адресного регистра R (рис. 1, *а*).

Информация с выхода адресного регистра R поступает на вход пирамидального дешифратора, который декодирует двоичное слово длиной n бит и устанавливает в единичное состояние одну из его выходных шин.

Матрица C состоит из системы горизонтальных и вертикальных проводников. Показанные на матрице C точки представляют собой вентили, связывающие эти две системы проводников. Таким образом, электрический сигнал, распространяющийся вдоль любого горизонтального проводника, в месте, где указана точка, пройдет через эту связь и будет распространяться по вертикальным проводникам.

Каждая вертикальная линия матрицы C соединена с одним из исполнительных элемен-

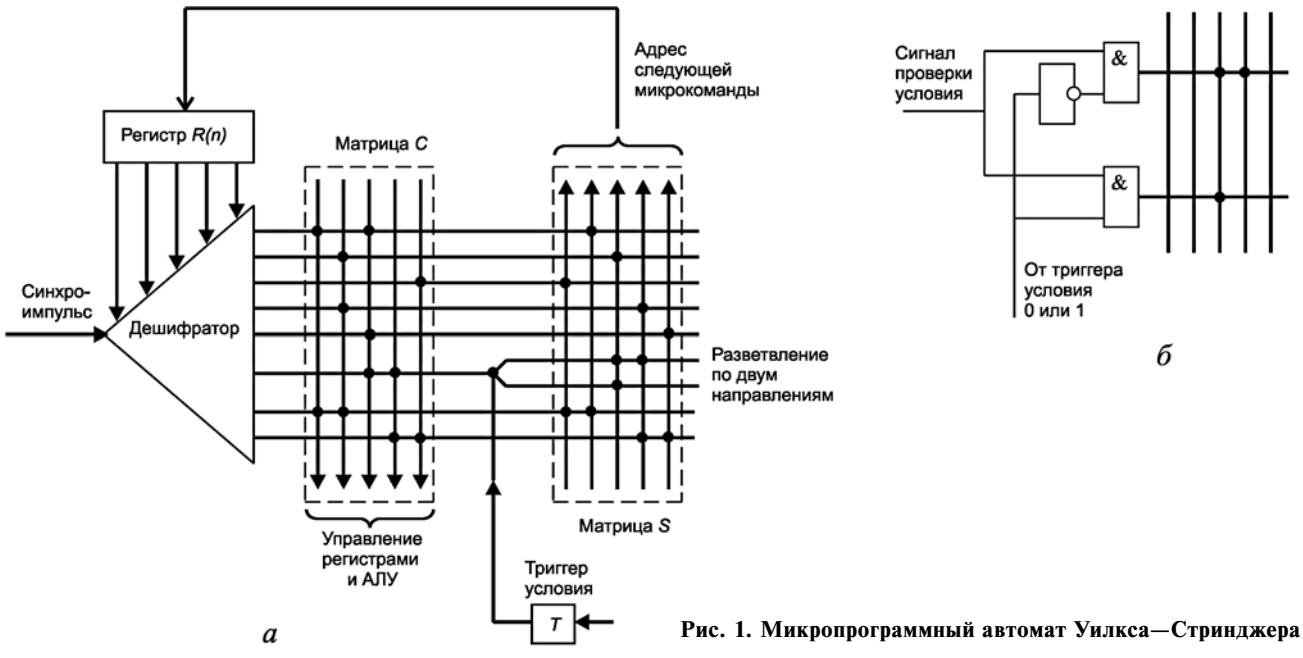

Рис. 1. Микропрограммный автомат Уилкса—Стринджера

тов управляемого устройства. Горизонтальную линию можно условно представить как микрокоманду. Когда эта линия возбуждена, она в свою очередь возбудит через вертикальные шины заранее заданный набор исполнительных элементов.

Импульс от дешифратора проходит также и сквозь матрицу *S*, которая задает последовательность действий. Выходные линии этой матрицы устанавливают соответствующий код в адресном регистре *R*. В соответствии с содержимым регистра *R* выбирается микрокоманда, которая выдает импульсы, управляющие исполнительными элементами в соответствии с программой, хранящейся в матрицах *C* и *S*.

В микропрограммном автомате Уилкса—Стринджера предусмотрена возможность ветвления микропрограмм, необходимая для условной передачи управления (рис. 1, *б*).

Синтез микропрограммных систем управления

Распространим принцип микропрограммного управления на дискретно-логические системы управления технологическими объектами. Назовем такого рода системы микропрограммными системами управления [4].

Микропрограммная система управления — это дискретный автомат, который в процессе функционирования генерирует последовательность микрокоманд, передаваемых в качестве управляющих воздействий на внешний объект управления.

Как и в микропрограммном автомате Уилкса—Стринджера, в микропрограммной системе управления микрокоманды запрограммированы и хранятся в виде двоичных кодов в матрицах *C* и *S* (рис. 2).

Особенности управления технологическими объектами не требуют сложной адресации микрокоманд. Они выбираются из матриц в порядке последовательной очереди. Поэтому в микропрограммной системе не требуется дешифратор адресов, а функции указателя микрокоманд выполняет *регистр микрокоманд*, состоящий из цепочки взаимодействующих триггеров *T*.

Микрокоманды из матрицы *C* передаются на вход объекта управления. Дискретные датчики *A*, *B*, *C*, *D*, *E* на выходе объекта управления формируют сигналы обратной связи, которые поступают на схемы **И** в матрице *S*.

Рассмотрим работу микропрограммной системы управления. Импульс пуска *P* включает первый (сверху) триггер регистра микрокоманд. Сигнал с его выхода возбуждает горизонтальную линию 1 матриц *C* и *S*. В матрице *C* с помощью вентилей или схем **ИЛИ**, присоединенных к этой линии, отбираются сигналы, которые в виде параллельного двоичного кода передаются на объект управления. После этого система переходит в режим ожидания конца исполнения переданной на объект управления микрокоманды.

Окончанием действия, заданного микрокомандой, служит сигнал с датчика *A*. Этот сигнал открывает схему **И**, включенную последовательно в первую линию матрицы *S*. В результате на выходе матрицы *S* формируется единичный сигнал, который по цепи обратной связи поступает в регистр микрокоманд и открывает в нем следующий по порядку (второй) триггер. Этот триггер, включившись, возбуждает горизонтальную линию 2 и одновременно сбрасывает вышестоящий первый триггер *T*. Таким образом, управление микрокомандами передается от первой горизонтальной линии ко второй, и далее процесс повторяется.

После отработки последней микрокоманды выключается конечный триггер регистра микро-

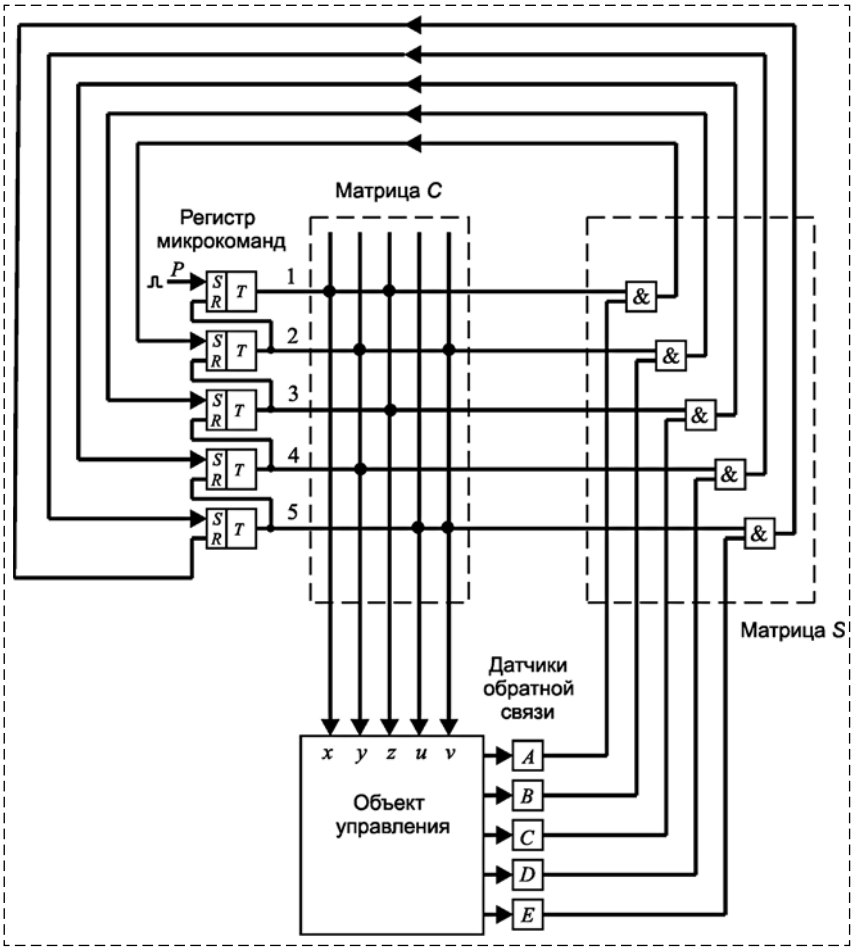

Рис. 2. Схема микропрограммной системы управления

команд, и система управления приходит в исходное состояние.

Главное отличие микропрограммной системы управления от известных многотактных систем [3, 4] состоит в том, что она не отслеживает состояния дискретного автомата. Последовательная выборка микрокоманд, запрограммированных в компьютере, позволяет управлять внешним объектом без применения дополнительных (внутренних) элементов памяти. Поэтому в процессе синтеза микропрограммных систем не требуется строить сложные циклограммы работы механизмов, исключать повторяющиеся состояния дискретного автомата и минимизировать логические функции. В результате проектирование дискретно-логических систем значительно упрощается.

Рассмотренная схема микропрограммной системы управления (рис. 2) — это лишь ее упрощенная модель, которую при реализации на компьютере необходимо преобразовать в рабочую функциональную схему. С этой целью используем язык функциональных блоковых диаграмм *FDB*, отвечающий требованиям международного стандарта *IEC*61131—3 и входящий в состав инструментальных систем программирования *ISaGRAF*, *CodeSys* и др.

С помощью этого графического языка представляем микропрограммную систему управления (рис. 2) в виде функциональной схемы (рис. 3).

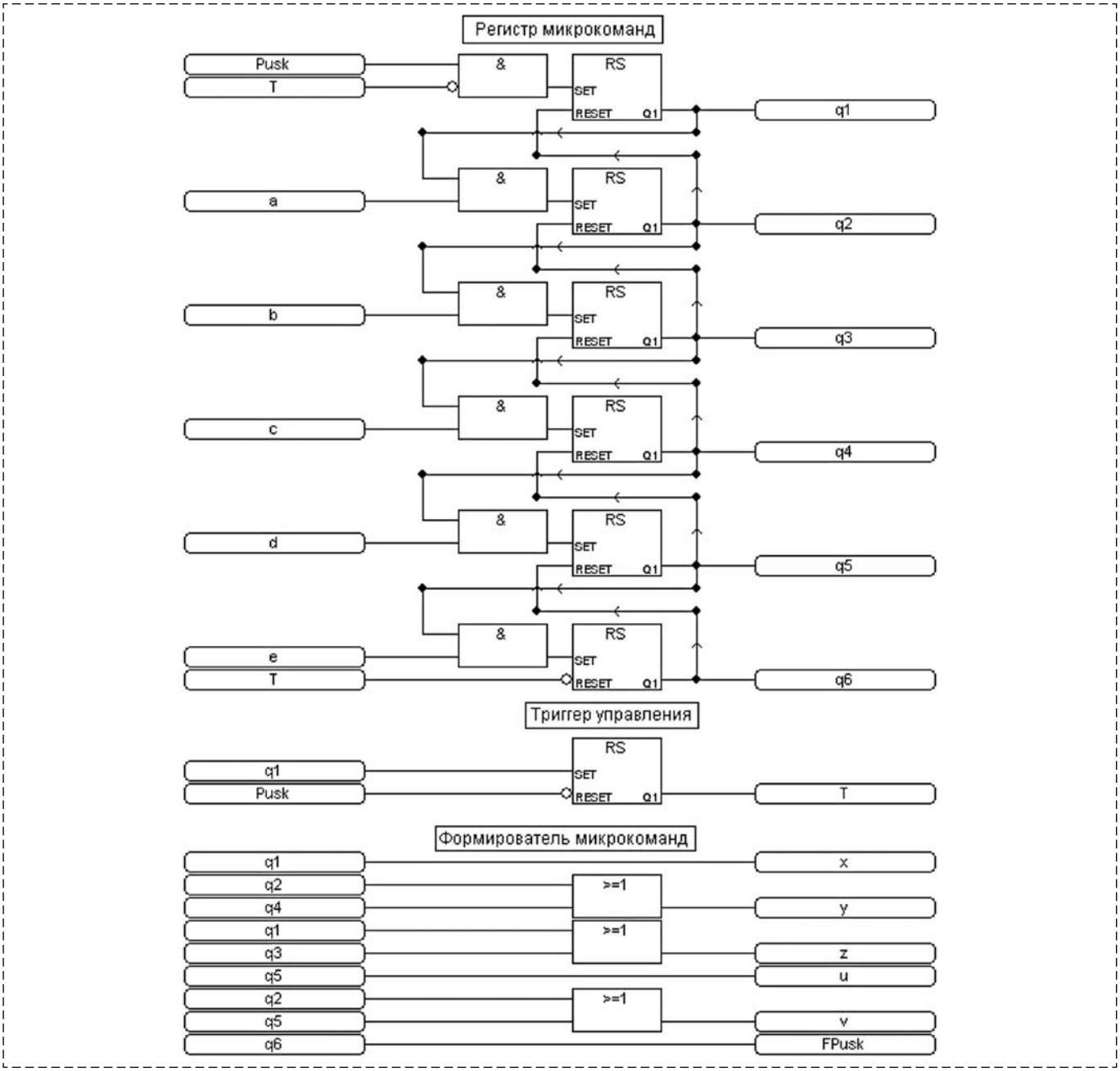

Рис. 3. Функциональная схема микропрограммной системы управления

Ее можно использовать как унифицированный шаблон для построения различных микропрограммных систем, работающих в разнообразных условиях. Логические связи в данной функциональной схеме аналогичны связям, которые рассматривались в прототипе микропрограммной системы управления (см. рис. 2).

Унифицированная программа (рис. 3) включает три компонента: регистр микрокоманд, триггер управления и формирователь микрокоманд.

Регистр микрокоманд для рассматриваемой системы состоит из шести статических *RS*-триггеров. Выходные сигналы этих триггеров $q1$, $q2$, $q3$, $q4$, $q5$, $q6$ объявлены в программе как локальные переменные. При переходе подсистемы из одной позиции в следующую позицию соответственно следующий триггер в регистре микрокоманд включается, а предыдущий выключается.

Триггер управления выполняет функции флага. Сигнал T на его выходе показывает, в каком состоянии находится система. Если $T = 0$, то это значит, что система находится в состоянии ожидания. Сигнал $T = 1$ свидетельствует о том, что система находится в работе.

Формирователь микрокоманд представляет собой простую комбинационную схему, состоящую в данном случае из логических элементов **ИЛИ,** которая выбирает микрокоманды с шины $q1$, $q2$, $q3$, $q4$, $q5$, $q6$ для передачи их на объект управления.

Рассмотрим функционирование программы. Сигнал *Pusk* = 1 поступает из подсистемы верхнего уровня и остается включенным во все время работы системы. Через элемент **И** он включает первый триггер регистра микрокоманд. Возбуждается линия $q1$, и включается триггер управления, который своим сигналом T блокирует дальнейшее воздействие сигнала *Pusk* на вход регистра микрокоманд. Включение триггера управления свидетельствует о том, что команда на включение системы принята, и система приступила к исполнению этой команды.

Одновременно в формирователе микрокоманд с линии $q1$ отбирается сигнал $x = 1$, который пересылается на объект управления для исполнения. По окончании действия, заданного микрокомандой, срабатывает датчик A объекта управления. Сигнал с этого датчика $a = 1$ включает второй триггер регистра микрокоманд. Возбуждается линия $q2$ на выходе этого регистра. Одновременно по цепи обратной связи выключается вышестоящий первый триггер. Подсистема переходит во вторую позицию. Далее работа программы циклически продолжается по рассмотренному алгоритму, пока не включится последний триггер регистра микрокоманд с выходным сигналом $q6$.

Заметим, что включение каждого нового сигнала qi предшествует отключению последующего сигнала $q(i - 1)$, что предотвращает прерывание выходных сигналов микропрограммной системы управления в моменты переключений триггеров в регистре микрокоманд.

Сигнал на последней линии $q6$ формирует сигнал *FPusk* = 1, который передается в систему верхнего уровня. Получив этот сигнал, система верхнего уровня гасит сигнал *Pusk* и передает управление некоторой другой подсистеме нижнего уровня.

Как только переменная *Pusk* примет значение *Pusk* = 0, триггер управления выключится. Инверсный сигнал с выхода этого триггера выключает последний триггер регистра микрокоманд, и на этом работа системы завершается.

Микропрограммная система управления механизмом автоматической смены инструмента

Изучим методику синтеза микропрограммных систем управления на примере механизма автоматической смены инструмента (МАСИ) станка с ЧПУ (рис. 4)

МАСИ имеет два автооператора (захватных устройства) ЗУ1 и ЗУ2. Первое захватное устройство ЗУ1, совершая вращательное и поступательное движения, переносит отработавший инструмент из шпинделя (Ш) в промежуточное загрузочное место (ПЗМ), а новый инструмент — из ПЗМ в Ш.

Вращательное движение ЗУ1 происходит с помощью электродвигателя Д, а поступательное — посредством гидравлического цилиндра Ц1. В шпинделе инструмент зажат пружиной. Для ее отжатия и освобождения инструмента служит цилиндр Ц2.

Второе захватное устройство ЗУ2 может совершать два поступательных движения — горизонтальное и вертикальное, причем в горизонтальном направлении положение механизма контролируется в трех точках (1, 2 и 3), а в вертикальном — в двух. Привод ЗУ2 гидравлический (цилиндры Ц3 и Ц4).

Подсистема 2, которая управляет захватным устройством ЗУ2, за время отработки всего цикла смены инструмента запускается дважды.

Последовательность работы МАСИ представим в виде условной записи:

1-я подсистема:
ЗУ1 (+90°); ← И1 → ; ЗУ1 (↓); ЗУ1 (+180°);
ЗУ1 (↑); → И2 ← ; ЗУ1 (−90°);

2-я подсистема:
ЗУ2 (1 → 3); ЗУ2 (↓); ЗУ2 (3 → 2);
ЗУ2 (↑); ЗУ2 (2 → 1);

3-я подсистема:
Работа магазина М (в данном примере подробно не рассматривается);

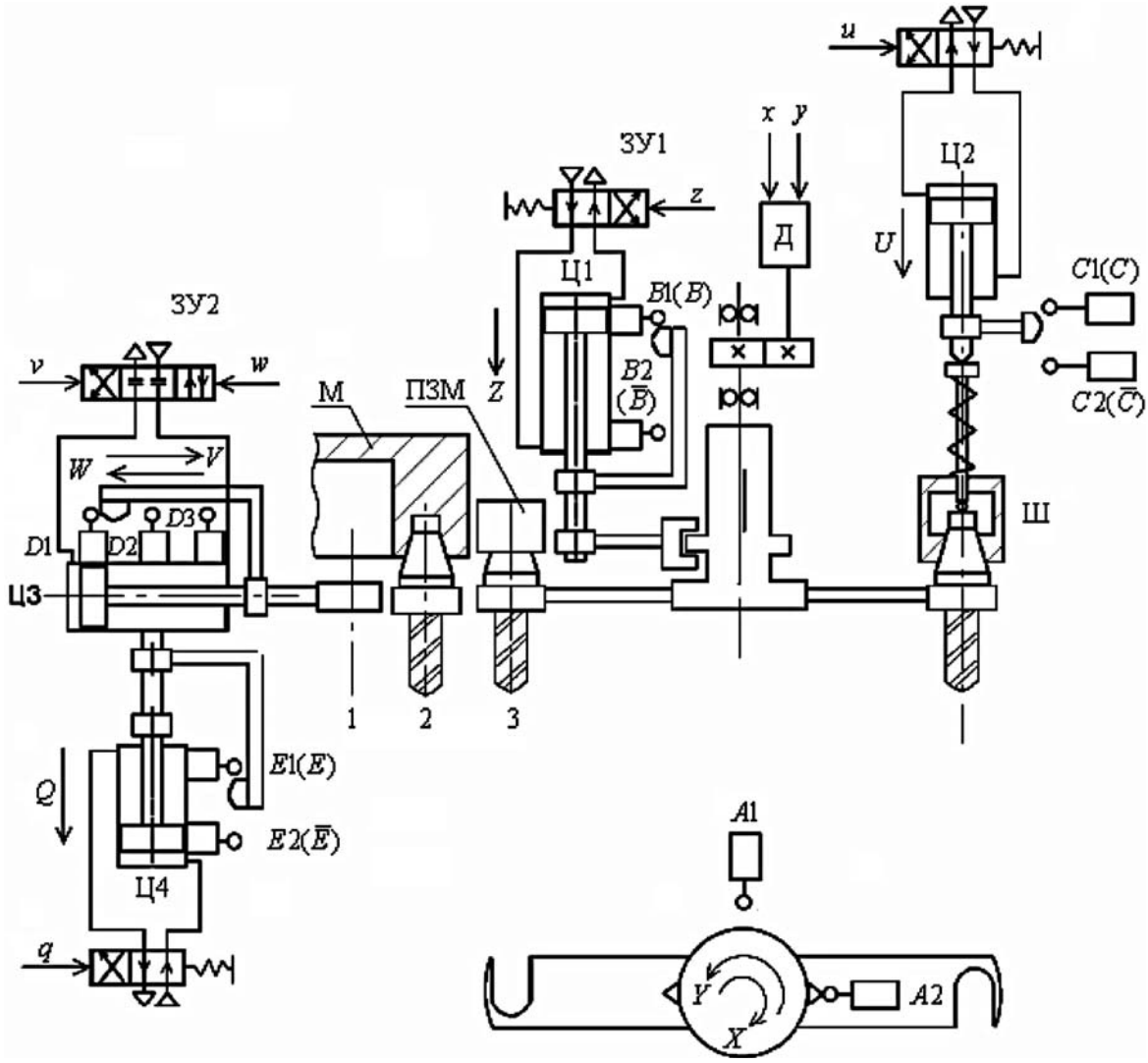

Рис. 4. Структурно-кинематическая схема устройства автоматической смены инструмента:
Д — двигатель; М — магазин; ПЗМ — промежуточное загрузочное место; Ш — шпиндель; (Ц1-Ц4) — гидравлические цилиндры

2-я подсистема:
ЗУ2 (1 → 2); ЗУ2 (↓); ЗУ2 (2 → 3);
ЗУ2 (↑); ЗУ2 (3 → 1).

Здесь стрелками показаны направления движения рабочих органов станка.

Смена инструмента в металлорежущих станках с ЧПУ программируется в управляющей программе (УП) с помощью технологических функций под именем *T*. Для исполнения этих функций служит система управления электроавтоматикой станка.

Если в очередном кадре УП задана функция *T*, то в системе управления электроавтоматикой специальный модуль технологических функций *TFUNK* пересылает значения технологических функций из массива данных системы ЧПУ в массив технологических команд. Одновременно модуль *TFUNK* заполняет массив управления драйверами *MDR*, в котором запоминаются заявки на включение драйверов для обслуживания периферийных устройств (рис. 5).

Ряд технологических функций требует ответа, подтверждающего исполнение указанных функций на станке. Для этой цели служит команда

MDR

1	*DRT*1	- Драйвер *T*
2	*DRT*2	- Драйвер *S*
3	*DRT*3	- Драйвер *M*
4	*OPROS*	- Драйвер опроса станка
5	*OTVET*	- Ответ со станка

Рис. 5. Массив управления драйверами *MDR*

OPROS, которая включает драйвер опроса станка. При получении утвердительного ответа (сигнал *OTVET*) со станка система ЧПУ гасит сигнал *OPROS*, а система управления электроавтоматикой, соответственно, выключает сигнал *OTVET*.

1. Синтез подпрограммы электроавтоматики S1

Рассмотрим диаграмму микрокоманд подпрограммы электроавтоматики *S*1 (рис. 6). Она содержит ряд условных переходов.

Диаграмма микрокоманд — это графическое изображение *позиций* и *микрокоманд*, соединенных *направленными линиями*. Позиции и микрокоманды чередуются. *Позиции* изображаются маленькими горизонтальными полосками, которые пересекают линии соединения. К каждой позиции присоединяются логические условия перехода к этой позиции. Возле каждой позиции указывается ее порядковый номер.

Микрокоманды изображаются в виде прямоугольников, внутри которых с помощью математических символов записываются действия системы управления во время исполнения данной микрокоманды. В специально отведенном поле можно задать *необязательный комментарий* — словесное описание действия системы управления, предписанное микрокомандой.

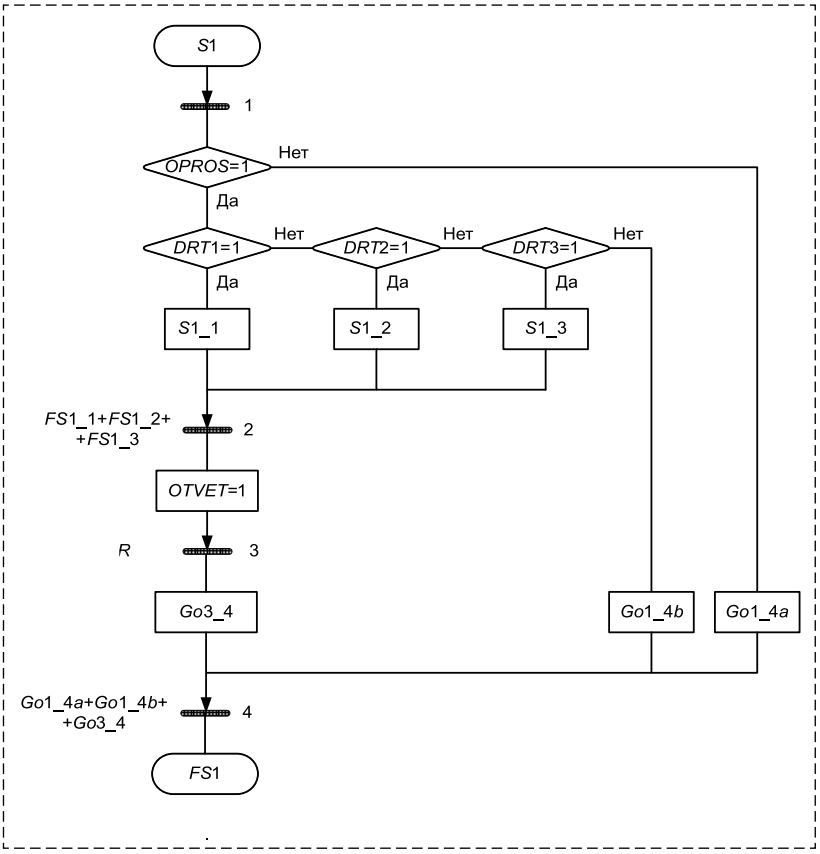

Рис. 6. Диаграмма микрокоманд управления электроавтоматикой

Диаграмма микрокоманд похожа на диаграммы *SFC*, которые в системах *ISaGRAF* или *CodeSys* программируются на языке высокого уровня (языке последовательных функциональных схем) *SFC*. Такая схожесть объясняется тем, что язык *SFC* наследует принцип управления внешними объектами от микропрограммных автоматов, первым из которых был микропрограммный автомат Уилкса — Стринджера.

Этот же принцип управления использован в сетях Петри, которые, в свою очередь, послужили основой для построения языка *SFC* [1].

Во всех указанных устройствах осуществляется заранее запрограммированная последовательная выборка команд, хранящихся в памяти компьютера. Естественно, структура и работа этих устройств разная, но главный принцип управления с помощью последовательного или адресного обращения к командам или микрокомандам их объединяет.

Рассматриваемая диаграмма микрокоманд принципиально отличается от диаграмм *SFC* тем, что может содержать сложные ветвления, условные и безусловные переходы, обращения к подпрограммам и др., а также содержит микрокоманды, которые непосредственно управляют объектом управления. Язык высокого уровня *SFC* такими возможностями не обладает. Программы на этом языке управляют объектом лишь с помощью вызова вспомогательных подпрограмм-посредников, созданных на основе одного из языков нижнего уровня. Чтобы выполнить тот или иной переход по условию, программа *SFC* также вынуждена обращаться к помощи подпрограмм нижнего уровня.

Расшифруем содержимое представленной диаграммы микрокоманд для подпрограммы *S*1. Сигнал *S*1 — это сигнал, разрешающий запуск системы управления в автоматическом цикле работы. Если *S*1 = 0, то подсистема управления находится в исходном состоянии. При поступлении сигнала *S*1 = 1 и при начальном значении сигнала с триггера управления $T = 0$ подсистема переходит в первую позицию, где с помощью условных переходов выбирается нужная ветвь программы.

Если сигнал *OPROS* = 0, то система управления электроавтоматикой никаких действий, связанных с работой драйверов, не выполняет. С помощью команды безусловного перехода *Go*1 − 4*a* = 1 регистр микрокоманд переводится с позиции 1 в позицию 3, из которой при логическом условии *Go*1_4*a* + *Go*1_4*b* + *Go*3_4 = 1

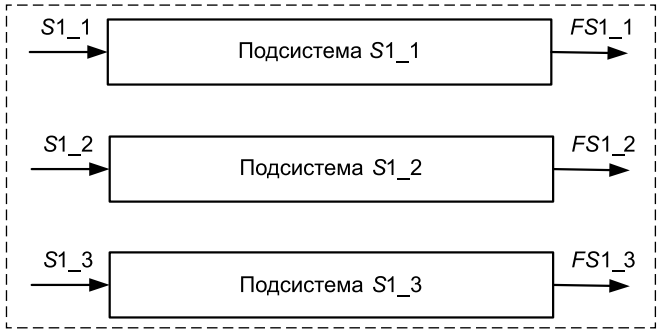

Рис. 7. Подпрограммы драйверов

система переходит в последнюю позицию 4, и работа программы заканчивается.

Если задано *OPROS* = 1, то подпрограмма электроавтоматики находит нужное имя драйвера и обращается к одной из подпрограмм *S*1_1, *S*1_2, *S*1_3, обслуживающих требуемое устройство станка (рис. 7).

Чтобы вызвать эти подпрограммы из подпрограммы электроавтоматики *S*1, достаточно установить в единичное состояние соответствующие входные переменные *S*1_1, *S*1_2 или *S*1_3. Во все время работы любой подпрограммы входная переменная обязательно должна находиться в единичном состоянии.

По окончании работы каждая из подпрограмм возвращает в систему электроавтоматики *S*1 единичные значения соответствующих выходных переменных *FS*1_1, *FS*1_2 или *FS*1_3.

По окончании работы выбранной подпрограммы регистр микрокоманд по логическому условию *FS*1_1 + *FS*1_2 + *FS*1_3 = 1 переходит в позицию 2, где инициируется переменная *OTVET* = 1, после чего ожидается гашение системой ЧПУ заявки *OPROS*. Когда эта переменная переходит в состояние *OPROS* = 0, регистр микрокоманд переводится из состояния 2 в состояние 3, где с помощью сигнала *Go*3_4 = 1 он переводится в позицию 4. Работа программы управления электроавтоматикой *S*1 завершается, и переменная *OTVET* становится равной 0.

Подпрограмма управления электроавтоматикой на языке *FBD* представляет собой функциональную схему (рис. 8).

Как и предыдущая функциональная схема (см. рис. 3), данная схема (рис. 8) включает в свой со-

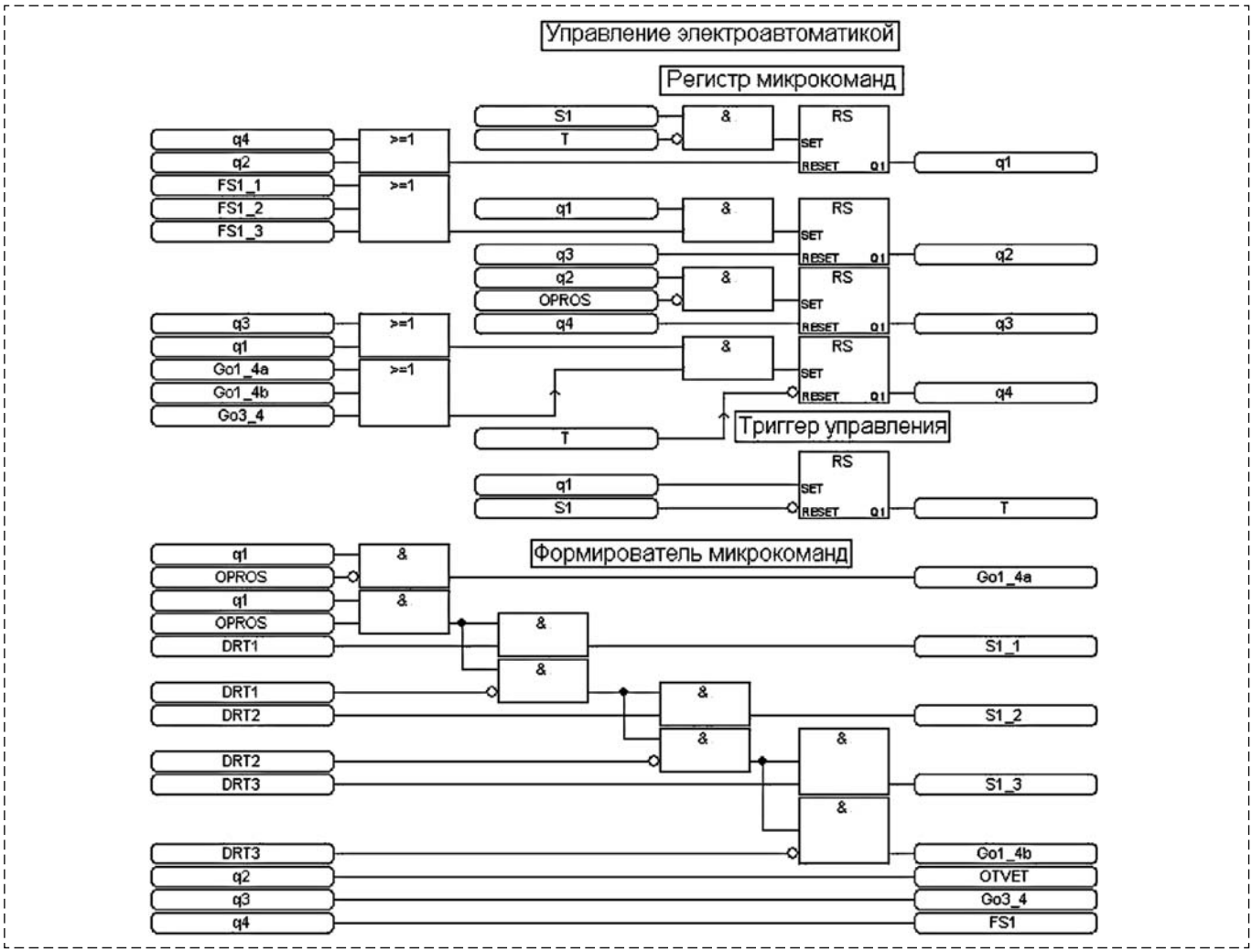

Рис. 8. Программа управления электроавтоматикой *S*1

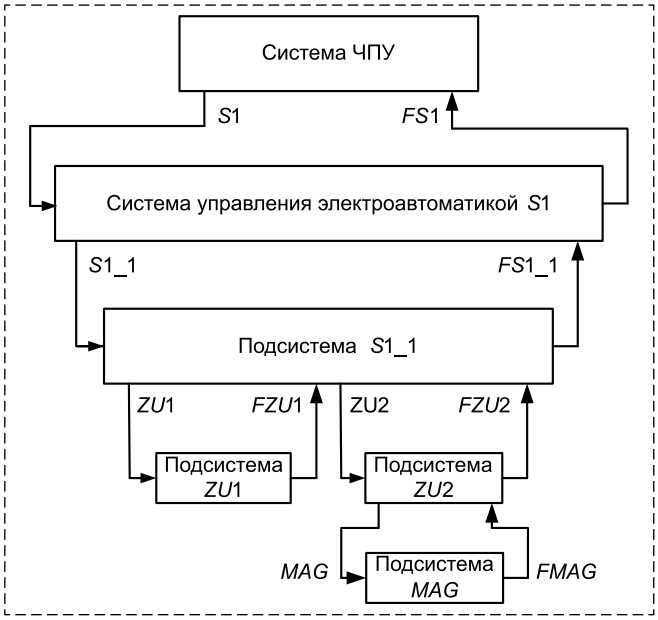

Рис. 9. Подсистема управления механизмом смены инструмента:
*ZU*1 — загрузочное устройство 1; *ZU*2 — загрузочное устройство 2; *MAG* — магазин

став три типовых блока: регистр микрокоманд, триггер управления и формирователь микрокоманд. Чтобы упростить прорисовку функциональной схемы, перекрестные связи в регистре микрокоманд показаны с помощью дополнительных ярлычков с именами переменных. Условные переходы программы реализованы в формирователе микрокоманд с помощью простой комбинационной схемы на элементах **И**.

Если переменные *OPROS* = 1 и *DRT* = 1, то по внешнему сигналу *S*1 = 1 включается первый триггер регистра микрокоманд. Сигнал *q*1 на его выходе принимает значение 1. Соответственно, в формирователе микрокоманд создается микрокоманда, которая запускает в работу дочернюю подпрограмму *S*1_1. По окончании ее работы на вход материнской подсистемы *S*1 поступает сигнал *FS*1_1 = 1, что приводит к переключению регистра микрокоманд с позиции 1 в позицию 2. В этой позиции по сигналу *q*2 = 1 формирователь микрокоманд присваивает переменной *OTVET* значение 1. Система ЧПУ, получив это сообщение, присваивает переменной *OPROS* значение 0. В результате регистр микрокоманд переходит в позицию 3, из которой с помощью микрокоманды безусловного перехода *Go*3_4 далее переходит в последнюю позицию 4, где формируется сигнал окончания работы подсистемы *FS*1 = 1.

Из диаграммы микрокоманд (см. рис. 6) видно, что подпрограмма драйвера *S*1_1 запускается из основной программы электроавтоматики *S*1 (рис. 9).

Подсистема управления механизмом смены инструмента *S*1_1, в свою очередь, включает две подсистемы: подсистему управления первым загрузочным устройством *ZU*1 и подсистему управления вторым загрузочным устройством *ZU*2.

Ставится задача синтезировать подпрограмму драйвера *S*1_1 для управления механизмом смены инструмента.

2. Синтез подсистемы S1_1

Подсистема *S*1_1 простая. Она последовательно вызывает две подпрограммы нижнего уровня: подпрограмму управления первым загрузочным устройством *ZU*1 и подпрограмму управления

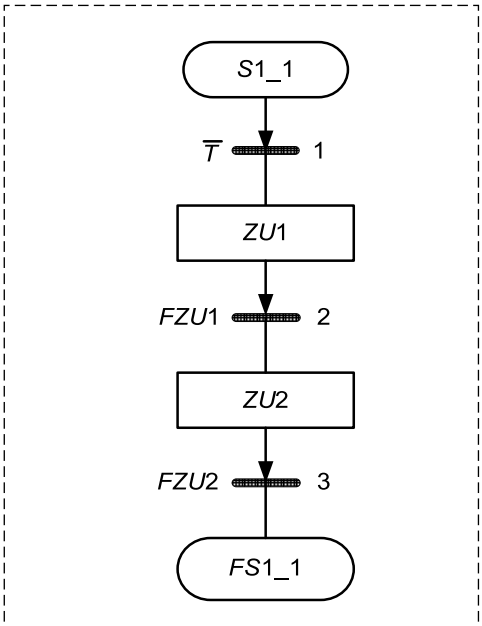

Рис. 10. Диаграмма микрокоманд подсистемы *S*1_1

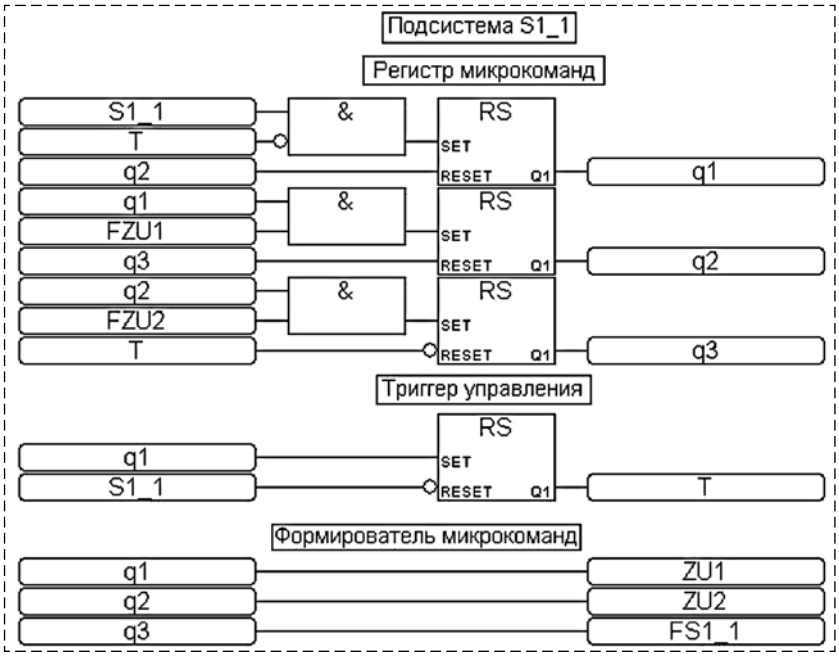

Рис. 11. Программа подсистемы *S*1_1

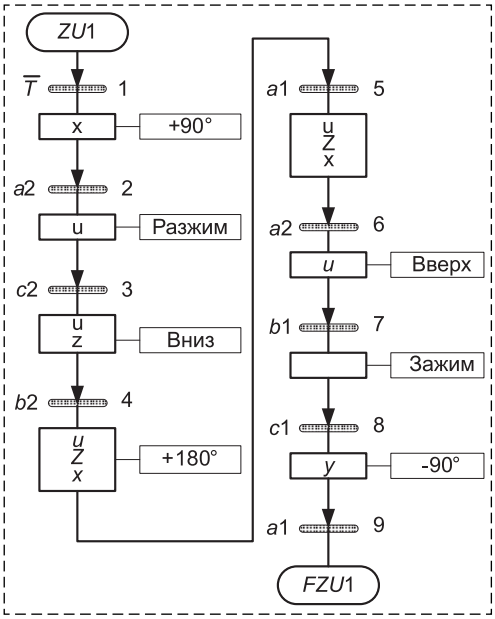

Рис. 12. Диаграмма микрокоманд подсистемы *ZU*1

вторым загрузочным устройством *ZU*2. Диаграмма микрокоманд подсистемы *S*1_1 состоит всего из трех позиций (рис. 10).

Программа подсистемы *S*1_1 также простая (рис. 11).

3. Синтез подсистем ZU1 и ZU2

Рассмотрим процесс проектирования подсистемы *ZU*1. Строим диаграмму микрокоманд подсистемы *ZU*1 (рис. 12).

Программа подсистемы *ZU*1 представлена на рис. 13. Методика построения микропрограммной подсистемы *ZU*2 аналогична.

Программы созданного проекта в *ISaGRAF* удобно представить в виде функциональных блоков [1], связанных в древовидную структуру (рис. 14).

Заключение

Главная отличительная особенность рассматриваемой методики программирования дискретно-логических систем состоит в том, что в программу *FBD* вносится один из основных элементов языка *SFC*, заимствованных из микропрограммных автоматов. Это процедура последовательной выборки микрокоманд из памяти компьютера, позволяющая расширить область решаемых дис-

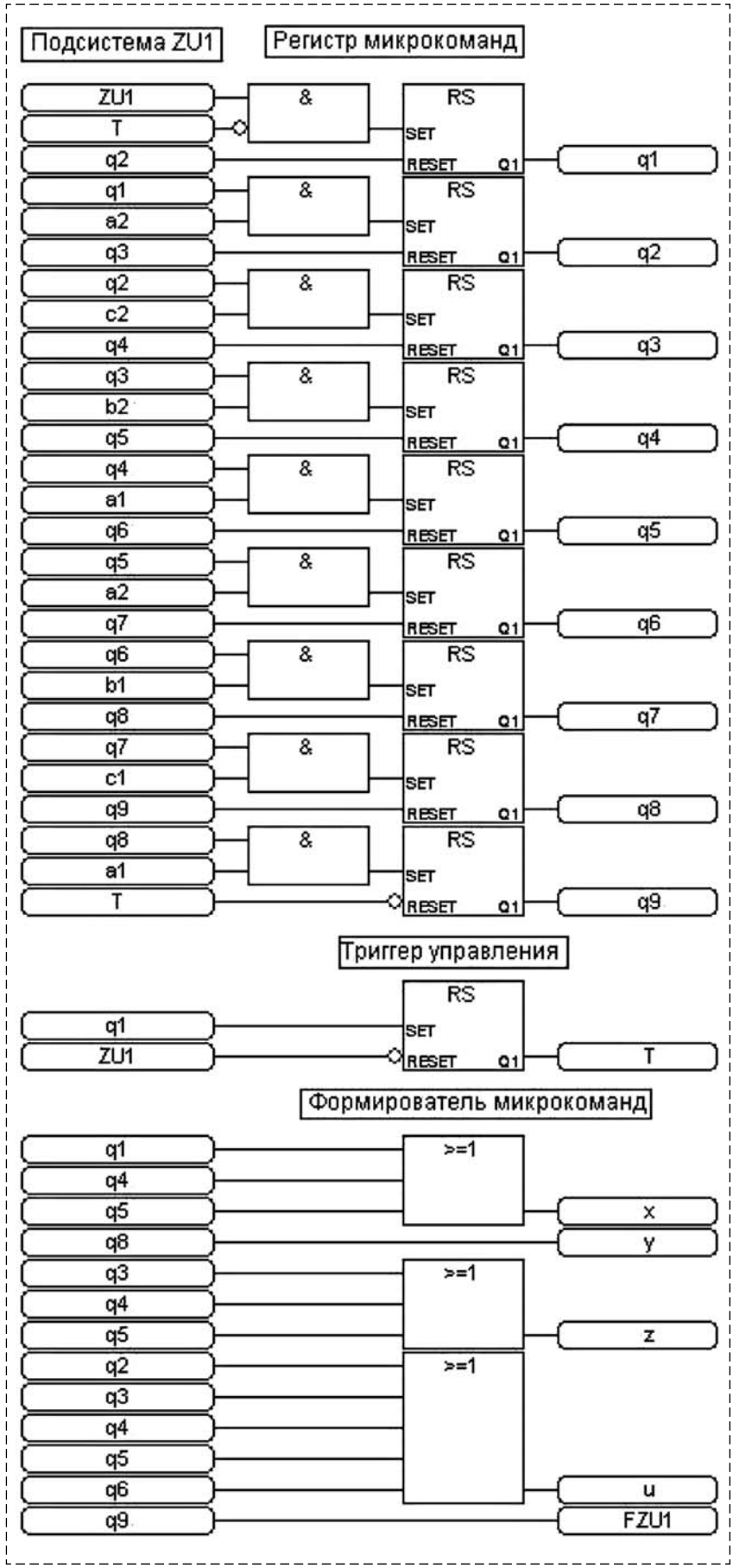

Рис. 13. Программа подсистемы *ZU*1

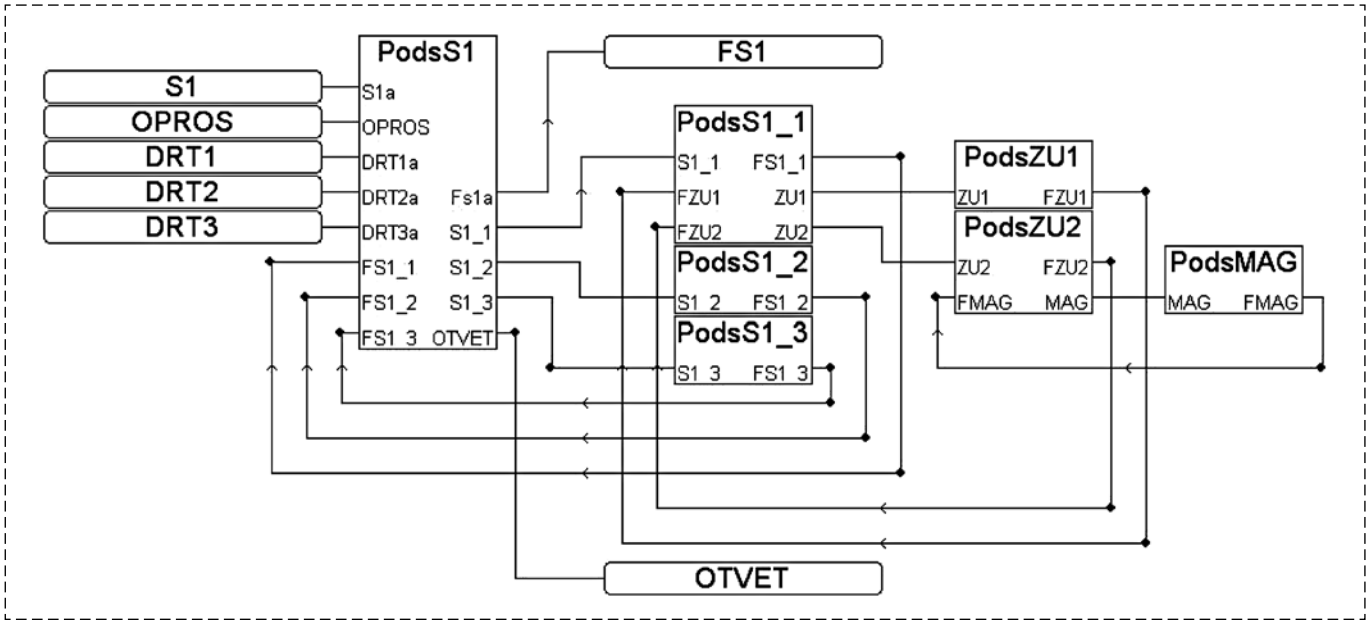

Рис. 14. Микропрограммная система управления электроавтоматикой с использованием функциональных блоков

кретно-логических задач на языке *FBD* и упрощающая их решение. Команды ветвления и обращения к подпрограммам существенно расширяют функциональность микропрограммных систем управления, создаваемых на языке *FBD*.

Практическая реализация микропрограммных систем управления сложными объектами, как и любых других систем управления, требует решения задач частного характера, возникающих в процессе проектирования. Графический язык функциональных диаграмм *FBD* хорошо подходит для решения такого рода вопросов, поскольку создаваемые в проекте *ISaGRAF* функциональные схемы наглядны, они отождествляются с реальными электронными схемами и поэтому сравнительно легко отлаживаются и тестируются с помощью средств инструментальной системы программирования. В конечном итоге в *ISaGRAF* функциональные схемы автоматически транслируются в прикладные программы, которые загружаются в ПЛК либо в промышленные компьютеры.

Созданные средствами *ISaGRAF* функциональные схемы можно также использовать для построения дискретно-логических систем управления на основе интегральных логических микросхем, включая ПЛИС. Кроме того, функциональные схемы могут служить в качестве промежуточных моделей для программирования дискретно-логических систем на языке С++.

Список литературы

1. **Петров И. В.** Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф. В. П. Дьяконова. М.: СОЛОН-Пресс, 2004. 256 с.

2. **Хассон С.** Микропрограммное управление. Вып. 1. М.: Изд-во Мир, 1973. 240 с.

3. **Чикуров Н. Г.** Логический синтез дискретных систем управления: учеб. пособ. Уфа: Изд. УГАТУ, 2003. 132 с.

4. **Чикуров Н. Г.** Синтез дискретно-логических систем управления: учеб. пособ. М.: ИНФРА-М, 2018. 229 с.

Synthesis of Microprogram Discrete Logic Control Systems

N. G. Chikurov, tchikurov@yandex.ru,
Ufa State Aviation Technical University, Ufa, 450000, Russian Federation,

Corresponding author: **Chikurov Nikolay G.,** Ph. D., Associate Professor,
Ufa State Aviation Technical University, Ufa,
450000, Russian Federation, E-mail: tchikurov@yandex.ru

Accepted on December 26, 2017

The article discusses the engineering synthesis of discrete-logic control systems for industrial machinery based on the algebra of logic. Given the method of designing new kinds of systems — Firmware discrete logic control systems. Projects of microprogram control systems are created in the instrumental ISaGRAF programming environment using the language of functional blocks FBD. An example of programming of the control system of tool electroautomatics in FBD is given.

The main distinguishing feature of the technique of programming discrete-logical systems is that one of the main elements of the high-level language SFS in included in the FBD program. This language, in turn, is taken from microprogramming machines and provides the procedure of consecutive sample of micro-ops from the memory of the computer that allows you to extend the scope of solvable discrete-logic problems in FBD and simplifies their solution. Branching commands and calls to subroutines greatly extend the functionality of microprogram control systems created in FBD. Practical implementation of microprogram control systems for complex objects, like any other management systems, requires solving problems of a private nature arising in the design process. Graphic language of functional diagrams FBD is well suited to solve this kind of issues because functional diagrams created in the ISaGRAF project are illustrative, they are identified with real electronic circuits and therefore they are relatively easily debugged and tested by means of the tool system programming. Ultimately, in ISaGRAF function diagrams are automatically translated into application programs that are loaded in the PLC or industrial computers. Function diagrams created by means of ISaGRAF, can also be used to build discrete-logic control systems based on integrated logic circuits. In addition, functional diagrams can serve as intermediate models for discrete programming-logical systems in the algorithmic language C++.

Keywords: *control, discrete-logic systems, electric circuits, machine tools*

For citation:

Chikurov N. G. Synthesis of Microprogram Discrete Logic Control Systems, *Mekhatronika, Avtomatizatsiya, Upravlenie,* 2018, vol. 4, no 19, pp. 232—242.

DOI: 10.17587/mau.19.232-242

References

1. **Petrov I. V.** *Programmable controllers. Standard languages and receptions of applied design*, Moscow, SOLON-Press, 2004, 256 p. (in Russian).

2. **Husson S.** *Microprogramming control.* Iss. 1, Moscow, Mir, 1973, 240 p. (in Russian).

3. **Chikurov N. G.** *Logicheskii sintez diskretnykh sistem upravleniya* (The logical synthesis of discrete control systems), Ufa, Publishing house of USAT University, 2003, 132 p. (in Russian).

4. **Chikurov N. G.** *Sintez diskretno-logicheskikh sistem upravleniya* (Synthesis of discrete-logic control systems), Moscow, INFRA-M, 2018, 229 p. (in Russian).