УДК 621-52

В.В.Ткачев, д-р техн.наук, С.Н., Проценко, Н.В.Козарь
(Украина, Днепропетровск, Национальный горный университет)

ФОРМАЛЬНЫЕ МЕТОДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ДИСКРЕТНОГО УПРАВЛЕНИЯ

Большинство объектов горной промышленности имеют логические законы управления, а все сигналы в системе управления носят дискретный характер. Предлагается формализовать процесс написания программ для таких систем, используя методы описания объектов управления, разработанные в теории цифровых автоматов. Фактически, задача дискретного управления объектами с использованием программируемых контролеров заключается в вводе входных сигналов, вычислении логической функции, связывающей входные и выходные сигналы, и выводе результата вычисления на выходы контроллера. Задачи дискретного управления можно записать в виде комбинационных и последовательностных автоматов. Задание условия для комбинационного автомата можно представить в виде таблицы истинности. Задачу построения комбинационного автомата удобно рассмотреть на примере разработки дешифратора для семисегментного индикатора. Входные сигналы для семисегментного индикатора поступают на разряды порта P1.0–P1.3, а управление индикатором осуществляется с порта 2 через буферный усилитель.

Решение.

Запишем таблицу истинности для дешифратора (табл.1).

Таблица 1

Входы				Выходы								Символ
1	2	3	4	5	6	7	8	9	10	11	12	13
P1.3	P1.2	P1.1	P1.0	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	1	1	0	1
0	0	1	0	0	1	0	1	1	0	1	1	2
0	0	1	1	0	1	0	0	1	1	1	1	3
0	1	0	0	0	1	1	0	0	1	1	0	4
0	1	0	1	0	1	1	0	1	1	0	1	5
0	1	1	0	0	1	1	1	1	1	0	1	6
0	1	1	1	0	0	0	0	0	1	1	1	7
1	0	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	0	1	1	0	1	1	1	1	9
1	0	1	0	0	1	1	1	0	1	1	1	A
1	0	1	1	0	1	1	1	1	1	0	0	B
1	1	0	0	0	0	1	1	1	0	0	1	C
1	1	0	1	0	1	0	1	1	1	1	0	D
1	1	1	0	0	1	1	1	1	0	0	1	E
1	1	1	1	0	1	1	1	0	0	0	1	F

Сформируем в памяти программ контроллера массив значений байта вывода согласно табл. 1 (метка BEGDIM), которые нужно вывести на индикатор для формирования шестнадцатеричных цифр от 0 до F. Текст подпрограммы, которая реализует дешифратор, приведен ниже.

```
DECOD:    MOV DPTR,#BEGDIM     ;заносим в DPTR адрес первого
                               ;элемента массива
          MOV A,P1             ; вводим входные сигналы
          ANL A,#0FH           ; маскируем старшие разряды входных
                               ; сигналов, получаем смещение в
                               ;таблице для данной комбинации
          MOVC A,@A+DPTR       ;заносим в аккумулятор значение
                               ;элемента массива соответствующего
                               ;входному коду
          MOV P2,A             ; выводим значение в порт
          RET                  ; выходим из подпрограммы
BEGDIM: DB 3FH, 06H, 5BH, 4FH, ; 0, 1, 2, 3
          DB 66H, 6DH, 7DH, 07H, ; 4, 5, 6, 7
          DB 7FH, 6FH, 77H, 7CH, ; 8, 9, A, B
          DB 39H, 5EH, 79H ,71H  ; C, D, E, F
```

В данной программе входное значение кода, образуемого входными сигналами, преобразуется в смещение для массива выходных значений. Затем с его помощью считываем из таблицы значение и выводим в выходной порт.

Такой способ реализации комбинационного автомата имеет следующие ограничения: входные сигналы должны быть подключены на один порт и подряд (иначе придется решать задачу сбора различных входов в одно входное слово); при достаточно большом количестве входов 7 и более и не полном использовании выходных комбинаций, нерационально используется память программ. Поэтому предлагается другой способ решения задачи управления, основанный на программной реализации логических коньюнктивно дизьюктивных уравнений.

Как и в предыдущем случае, функционирование автомата задается в виде таблицы истинности, например табл. 2.

Таблица 2

Входы				Выход
P1.2	P1.4	P2.1	P2.3	P1.7
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1

Запишем логическое уравнение для выхода согласно приведенной таблице.

P1.7=!P1.2 & P1.4 & P2.1 & P2.3 +P1.2 & !P1.4 & P2.1 & !P2.3+P1.2 & !P1.4 & P2.1 & !P2.3

Составим схему алгоритма для первой функции «И» уравнения (см. рис.1).

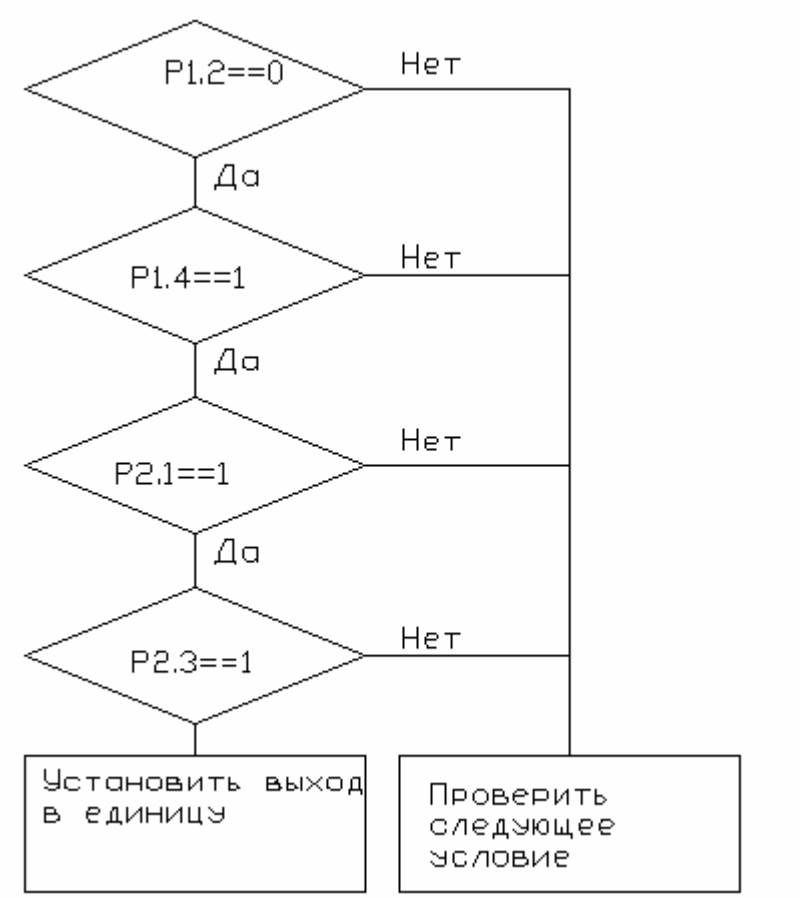

Рис.1. Программная реализация функции «И»

Программная реализация для этой функции имеет вид:

```
        JB P1.2,M1
        JNB P1.4,M1
        JNB P2.1,M1
        JNB P2.3,M1
        SETB P1.7
        LJMP EXIT
M1:     * * *           ;проверка следующего условия
```

Следует обратить внимание, что команде перехода по условию соответствует условный оператор на блок-схеме алгоритма с результатом «нет» внизу. Поэтому для того, чтобы программа была читабельней, предлагается использовать команды, которые проверяют инверсное условие. Тогда последовательное выполнение таких команд и будет соответствовать выполнению исходного условия логической функции.

Схема алгоритма для функций «ИЛИ»приведена на рис.2.

В этом алгоритме под &1, &2 подразумеваются логические функции «И» по строкам таблицы приведенных уравнений. Полная программа логического уравнения, соответствующего таблице истинности, имеет вид:

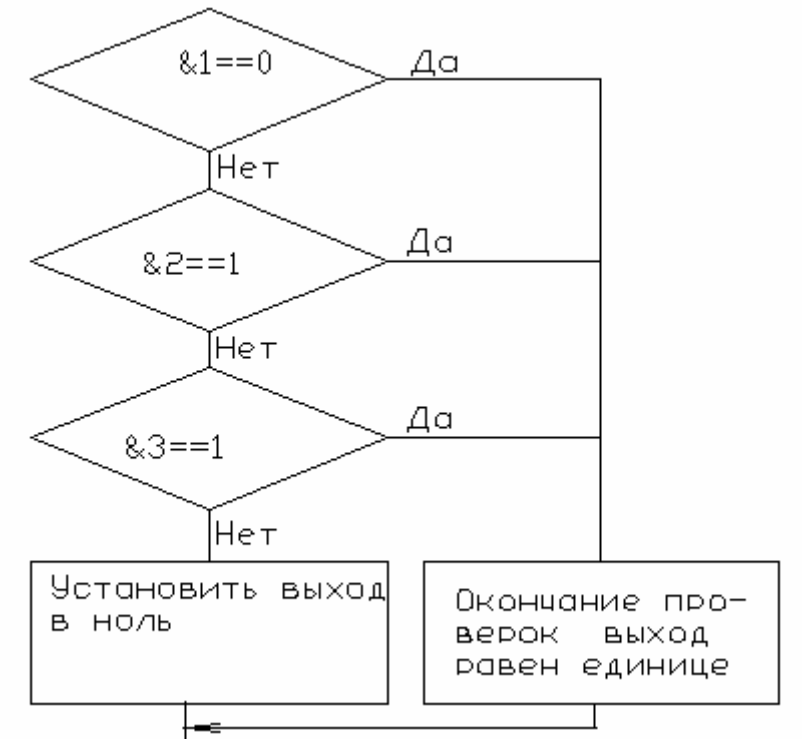

Рис. 2. Схема алгоритма для функций «ИЛИ»

Полная программа комбинационного автомата приведена ниже.

```
        JB    P1.2,M1       ; проверка условия первой строки таблицы
        JNB P1.4,M1
        JNB P2.1,M1
        JNB P2.3,M1
        SETB P1.7           ; установка выхода в единицу
        LJMP EXIT           ;и выход
M1:
        JNB P1.2,M2         ;проверка условия второй строки таблицы
        JB    P1.4,M2
        JNB P2.1,M2
        JB    P2.3,M2
        SETB P1.7           ;установка выхода в единицу
        LJMP EXIT           ;и выход
M2:
        JNB P1.2,M3         ;проверка условия третьей строки таблицы
        JB    P1.4,M3
        JNB P2.1,M3
        JB    P2.3,M3
        SETB P1.7           ; установка выхода в единицу
        LJMP EXIT           ;и выход
M3:     CLR    P1.7         ;сбросить выход в ноль – не выполнилось ни одно
                            ;из условий
EXIT:   RET
```

Алгоритмы управления последовательностных автоматов или автоматов с памятью удобно записывать в виде направленных графов. Теория направленных графов подробно рассмотрена в работе [1]. В предложенной методике на-

писания программ дискретного управления, вершины графа представляют собой состояние системы дискретного управления, дуги указывают направление перехода. Над дугами записываются условия перехода из одного состояния в другое, а через слеш (/) указываются выходные сигналы, которые при этом формируются. Функция управления в этом случае вычисляется в виде логического уравнения, в котором к входным сигналам добавлены еще значения состояния объекта. Программный способ реализации автомата с памятью значительно расширяет список используемых входных и выходных переменных. Кроме привычных дискретных переменных, можно использовать: численные переменные, счетчики, временные задержки, значения АЦП, выходные значения для ШИМ и т.д. Операции, которые можно выполнять над численной переменной, используемой в качестве выходной: присваивание значения, сброс в 0, арифметические действия. Над численными входными переменными выполняются операции сравнения на равенство, больше, меньше, проверка на ноль. В качестве примера численной переменной может быть переменная, которая хранит состояние объекта. Счетчики могут быть как входными, так и выходными переменными. Над счетчиками – выходными переменными можно выполнять операции инкрементирования, декре-ментирования, сброса в ноль, установки в заданное значение. Над счетчиками – входными переменными можно выполнять операции сравнения на больше, меньше, на ноль или нужное значение. Таймеры фактически также являются счетчиками, но тактированными по отметчику времени. Как выходной сигнал таймер можно запустить, остановить, установить на заданное время. При использовании таймера в качестве входного сигнала, его можно контролировать на ноль.

Рассмотрим пример автоматизации упрощенной шахтной водоотливной установки. Водоотливная установка состоит из зумпфа, где собирается вода, и двух насосов – центробежного (основного) и заливочного. Контроль за уровнем воды в зумпфе осуществляется двумя датчиками: верхнего и нижнего уровня.

Схема объекта управления приведена на рис.3. Алгоритм функционирования водоотливной установки следующий: если вода находится ниже отметки уровня верхнего датчика, оба насоса выключены. При достижении водой отметки уровня верхнего датчика включается заливочный насос. Через 10 с работы заливочного насоса включается основной насос и выключается заливочный. Основной насос работает до тех пор, пока вода в зумпфе опустится ниже отметки уровня нижнего датчика. После чего основной насос отключается и система переходит в исходное состояние.

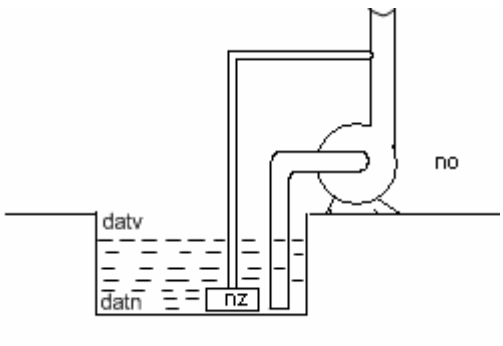

Рис 3. Объект управления

Составление графа алгоритма управления:

Выделим состояния, в которых может находиться система и граф.

0–оба насоса выключены, вода находится ниже отметки уровня верхнего датчика.

1– включен заливочный насос, производится заливка основного насоса.

2– производится откачивание воды из зумпфа.

Определение условий перехода из одного состояния в другое.

Система переходит из состояния 0 в состояние 1 тогда, когда уровень воды достигнет датчика верхнего уровня (выходное значение датчика верхнего уровня равно 1).

Система переходит из состояния 1 в состояние 2 после окончания 10 с работы заливочного насоса.

Система переходит из состояния 2 в состояние 0 при снижении уровня воды в зумпфе ниже датчика нижнего уровня.

Определение выходных сигналов при переходе из состояния в состояние.

При переходе системы из состояния 0 в состояние 1 включается заливочный насос и устанавливается выдержка времени равная 10 сек.

При переходе системы из состояния 1 в состояние 2 выключается заливочный насос и включается основной насос.

При переходе системы из состояния 2 в состояние 0 выключается основной насос.

Граф состояния водоотливной установки приведен на рис. 4 . Условия перехода и выходные сигналы здесь нанесены на дуге перехода через слеш.

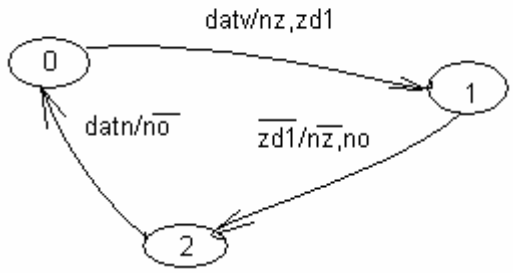

Рис.4. Граф состояния системы управления

Программа, реализующая граф на языке ассемблер, для контроллера MCS51 приведена ниже

```
;*************************************************************
;Программа управления водоотливной установкой с заливочным насосом
;*************************************************************
datn    equ 81h             ;битовый адрес датчика нижнего уровня
datv    equ 80h             ;верхнего уровня
no      equ 90h             ;основного насоса
nz      equ 91h             ;заливочного насоса
zd1     equ 31h             ;ячейка формирования задержки
state   equ 40h             ;ячейка состояния автомата
```

```
prescal equ 30h                 ;ячейка предделителя

        ljmp m0                 ;старт с обходом векторов прерывания

        org 0bh                 ;вектор прерывания от таймера TM0
        ljmp tim0

m0:     clr no                  ;начало программы
        clr     nz              ; сброс выходов в исходное состояние
        mov     state,#0        ;состояние автомата 0
        mov     sp,#70h         ;стек в 70
        mov     tmod,#1h        ;установка таймера 0 в режим 1
        mov     th0,#3ch        ;период вызова таймера 50 мс
        mov     tl0,#0b0h
        setb tcon.4             ;разрешение работы таймера
        setb ie.7               ; общее разрешение прерывания
        setb ie.1               ;разрешение прерывания от таймера
        mov prescal,#20         ; начальная установка предделителя

 loop:  mov a,state             ; основной цикл автомата
        cjne a,#0,jst1          ; определение номера состояния
        jnb datv,loop           ;проверка условия перехода в состоянии 0
        mov zd1,#1eh            ;условие выполнено
        setb nz                 ; включение выходных сигналов
        mov state,#1            ;переход в состояние =1
        ajmp loop               ;в начало цикла автомата

 jst1:  cjne a,#1,jst2          ;определение номера состояния
        mov a,zd1               ;проверка условия перехода в состоянии 1
        jnz loop                ;условие задержка =0
        clr nz                  ;выдача выходных сигналов
        setb no
        mov state,#2                    ;переход в состояние =2
        ajmp loop

jst2:   jb datn,loop            ;проверка условия перехода в состоянии 2
        clr no                  ;выдача выходных сигналов
        mov state,#0                    ;переход в сотояние=0
        ajmp loop
tim0:   push acc                ;обработчик прерываний от таймера 0
        mov th0,#3ch            ;занесение в таймер значения счета
        mov tl0,#0b0h
        djnz prescal,m2         ;декрементирование предделителя
        mov prescal,#20         ;установка нового значения предделителя
        mov a,zd1               ;проверка на 0 задержки
        jz m2
        dec zd1                 ;если не 0 декрементируем
  m2:   pop acc
        reti                            ;возврат из прерывания
        end
```

Реализация графа на языке С для PIC контроллеров имеет вид:

```
static void interrupt isr(void)                 // обработчик прерывания от таймера
{       if(TMR1IF == 1)
        {       TMR1ON = 0;
                TMR1IF = 0;
                TMR1H = 0x85;
                TMR1L = 0xed;
                TMR1ON = 1;
                if(--preskal == 0)
                {preskal = 4;
                        if(zd1 != 0)    --zd1;

                }
        }
}
void main()                                     //основная часть программы
{   Init();                                     //инициализация периферийных устройств
        for(;;)
        { Zadach1();                            //задача управления водоотливной установкой
                }
Void
Zadach1(void)
{       switch(state1)
                {case 0:                                        //состояние 0
                        {if(dV)                                  // проверка условия перехода в состоянии 0
                                {
                                  ZNo = 1;      //установка выходных сигналов
                                  zd1 = 30;
                                  state1 = 1;   //переход в состояние 1
                                    }
                                }
                                break;
                case 1:                                         //состояние 1
                        {if(zd1 == 0))                           // проверка условия перехода в состоянии 1

                                {ZNo = 0;                        //установка выходных сигналов
                                 NO = 1;
                                 zd2=600;
                                 state1 = 2;                     //переход в состояние 2
                                   }
                            }
                                break;
                case 2:                                         //состояние 2
                        {if(!dN)                                // проверка условия перехода в состоянии 2

                                {
                                   NO = 0;
                                   state1 = 0;
                                     }
                            }
                                 break;
                 }
    }
```

Достоинством данного подхода при написании программы управления является то, что процесс создания программного обеспечения практически полностью формализован. Программы, написанные по этой методике, четко структурированы и прозрачны для понимания. Граф понятен специалистам по проектированию систем и технологам, а поэтому исключаются ошибки, связанные с непониманием работы объекта. Отладка программного обеспечения очень наглядна, особенно, если в качестве «помощника» использовать цифровой индикатор, отображающий номер состояния графа. Программа вычисления управляющей функции линейна, т.е. не имеет циклов ожидания, а это значит, что задачи управления, оформленные в виде подпрограмм, легко решать в многозадачном режиме, используя только циклический метод планирования задач. При управлении сторожевым таймером упрощается проверка функционирования отдельных задач, для этого достаточно удостовериться в соответствии входных сигналов для текущего состояния задач.

1.Автоматизация процессов подземных горных работ./ Под ред. А.А. Иванова. – К.: Выща шк,1987. – 328 с.