Environment. Technology. Resources, Rezekne, Latvia
Proceedings of the 10th International Scientific and Practical Conference. Volume I, 67-71

Discrete automatic schemes for ASC TP

Vladimir Konevtsov, Igor Poletaev, Sergey Verteshev
FSG-fEIHPE «Pskov State University», Faculty of Informatics.
Adress: pl. Lenin, 2, Pskov, 180000, Russia.

Abstract. **The article given shows functional possibilities of creating discrete automatic schemes in CAD of digital automatic control system (CAD of digital ACS), in complex of Software Design of System of the Digital Control (Complex SDSDC) for automated system of control of technological processes (ASC TP). Possibilities of complex SDSDC for implementing combination and sequential control compared with requirements of international standard IEC 61131-3:2003 (Part 3: Programming languages) are estimated.**

Keywords: **comparator of vectors of logical signals, flip-flop, shift register, counter of clock periods, recognition of the front of a logical signal, vote function, decoder, coder, transformer of bit sequence in a vector of logical signals and visa versa, index register, timer, multiplexer, signal generator.**

I INTRODUCTION

For designing blocking systems, protection of technological equipment from overloading, technological mode control (included start and stop modes, switching under usual conditions and abnormal ones) discrete automatic schemes are used. These are realized by technical means according standard IEC 61131-1:2003 (Part 1: General information) programmable logic controllers (PLC). For mathematical description of these means system of calculation of expressions-Boolean algebra is taken [10, 14]. Logical control units are classified as combinational (having no built-in memory- automat without memory) and sequential ones (automat with memory). According to this classification one can differ units of single clock period (combinational) and multi clock period (sequential) control [1, 3, 5-7].

II ARCHITECTURE OF DISCRETE AUTOMATIC DEVICES

Any logical variable and function of binary logic is defined on multiple values {0, 1} or {false, true}. The function of binary logic can be single-place, two-place and multi place. Functions of repeat $Y = X$, negation $Y = \overline{X}$, constant-false $Y = 0$, constant-true $Y = 1$ belong to single placed functions. For two independent variables one can get 16 different functions [10, 14], which can be equivalently expressed through NOT function and conjunction function (AND) or through NOT function and disjunction function (OR). Along with these functions the function of addition on module two is used (XOR). These four functions are defined in the set of operations of expression calculation modules of complex SDSDC [8, 9, 13, 18, 19]. These are basic functions for implementation of discrete automatic schemes to perform memorizing, storing, counting, analyzing, comparing logical signals and vectors of logical signals. These devices are flip-flops, registers, counters, decoders, coders, commutators, comparators of circuit computer technology [10, 14] etc. Standardized functional modules of complex SDSDC are defined by analysis mathematical methods of control theory [8]. Comparative evaluation of features of complex SDSDC [13] and functional requirements IEC 61131-3 [16] is given in the table I [4, 12, 15, 17, 20].

Comparing vectors of logical signals (for example, finding a state of tens and hundreds discrete performing mechanisms on the state of end switchers in ASC TP) can be carried out more conveniently with comparator [14]:

$$Y_i = \begin{cases} X_i & if\ f=0 \quad reiteration \\ \overline{X_i} & if\ f=1 \quad negation \\ X_i \oplus Z_i & if\ f=2 \quad exelusiving\ OR \\ X_i \wedge Z_i & if\ f=3 \quad conjuction \\ X_i \vee Z_i & if\ f=4 \quad disjunction \end{cases}, \quad (1)$$

where $i = 1,2,\ldots,n$; n – is a signal vector dimension, f – is a parameter of choice of operation, X_i, Z_i – are components of vectors of dimension n.

TABLE I
COMPARATIVE EVALUATION OF FEATURES OF COMPLEX SDSDC

Groups of discrete automatic modules	Complex SDSDC	IEC 61131-3
Comparators of vectors of logical signals	+	-
Flip-flops	+	+
Shift registers	+	-
Counter of clock periods	+	+
Recognition of the front of logical signals	+	+
Vote functions	+	-
Decoders	+	-
Coders	+	-
Transformer of bit sequence in a vector of logical signals	+	-
Transformer of vector of logical signals in a bit sequences	+	-
Index registers	+	-
Timers (TP-Pulse, TON-ON delay, TOF-OFF delay)	+	+
Multiplexers	+	-
Signals generators	+	-

Note to the table I: (+) is presence, (-) is absence of standardized module group.

For storage of values of one signal of a logical look the RS and SR triggers [6] having two stable conditions in case of reset or set [10] (bistable elements) are used.

These flip-flops have a reset input R, a set input S, an input of a state and an inverse output of a state. For storing a vector from n logical signals one can define a «packet» RS flip-flops (this packaging is used by manufacturing integrated circuits [11]) and at programming it can be done very simply: at $n = 1$ flip-flop is (becomes) usual [10]:

$$\left.\begin{array}{l} Q_i(k)=\begin{cases} 0 & if\ R_i=1 \\ 1 & if\ S_i=1 \\ Q_i(k-1) & else \end{cases} \\ Y_i(k)=\overline{Q_i(k)}, \end{array}\right\}, \quad (2)$$

where i=1,2,…,n; n – is the number of flip-flops in «the packet». The signals having close inertance, i.e. having the same rate of change are included in «the packet» of logical signals. If the vector of logical signals is to contain signals of different inertance then the time required to process the scheme to have a block of this functional module is defined by the signal with the highest rate of changing its value. At the output of a flip-flop state with the reset priority when R=S=1 at the same clock period the value of the signal always equal to 0. At the output of a flip-flop state with the set priority when $R = S = 1$ at the same clock period the value of the signal always equal to:

$$\left.\begin{array}{l} Q_i(k)=\begin{cases} 1 & if\ S_i=1 \\ 0 & if\ R_i=1 \\ Q_i(k-1) & else \end{cases} \\ Y_i(k)=\overline{Q_i(k)}, \end{array}\right\}, \quad (3)$$

where i=1,2,…, n; n – is the number of flip-flops in «the packet». In the classical definition of RS and SR flip-flops the memory at the state input in a previous clock period is not accessible, this is an inner connection of a module, it is destroyed if power is switched off. In definitions of complex SDSDC this connection is accessible for an system engineer, and the memory at the input state in the previous clock period can be stored when power is switched off.

For storing signals of different types multi place registers are used («cells» of memory), they are registers of different action: series input-series output, series input-parallel output, parallel input- parallel output, parallel input-series output. In systems with memory (fixing the trend of pre-abnormal situation, correcting devices of high order, identification and imitation of systems etc.) registers with multi-value memory are used. These registers are referred to as shift registers [2, 8]:

$$\left.\begin{array}{l} Z2=\begin{cases}1 & if\ (q\geq N)\wedge(Z0=0)\\ 0 & if\ (q<N)\vee(Z0=1);\end{cases}\\ q=\begin{cases}q_0+1 & if\ (q<N)\wedge(Z1=1)\wedge(Z0=0)\\ N & if\ (q\geq N)\wedge(Z0=0)\\ 0 & if\ Z0=1;\end{cases}\\ Y_k=\begin{cases}X_k & if\ (Z1=1)\wedge(Z0=0)\\ X_{k-1} & if\ (Z1=0)\wedge(Z0=0)\\ 0 & if\ Z0=1;\end{cases}\end{array}\right\}, \qquad (4)$$

where $k=1,2,\ldots,N$; N – is the length of the trend; q_0 – is the initial length of the trend; $Z0$ – is the reset of the register; $Z1$ – is the shift of the register; q – is the current length of the register; *X(k), X(k-1),..., X(k- N)* – are values of the variable at the clock periods *(k*-1), (*k*-2), …, (*k-N*); y*(k), Y(k*-1*)*,…, Y*(k- N)* – is the content of the register.

Counters differ by the way of counting (direct, reverse), by representing in notation system (binary, decimal), by the way of count controlling (synchronous, asynchronous). The counter («the packet» counter) performs a relative count of time, that is count in clock periods of control scheme. The signals having close inertance, that is having the same rate of changing are included into «the packet» if signals of different inertence are to be connected to «the packet» of counters then the time required to process the scheme containing blocks of this functional module is defined by the signal of the highest rate of changing its values:

$$\left.\begin{array}{l} Z_i=\begin{cases}1 & if\ TZ_i=TT_i\\ 0 & else\end{cases}\\ Y_i=\begin{cases}TH_i & if\ C_i=1\\ TT_i+1 & if\ TT_i<TZ_i\\ TT_i-1 & if\ TT_i>TZ_i\\ TZ_i & if\ TT_i=TZ_i,\end{cases}\end{array}\right\} \qquad (5)$$

where C_i – is the reset of the counter; $i=1,2,\ldots,n$; n – is the number of counters in «the packet»; TH_i – is the initial value of the counter *i* at resetting; TZ_i – is the task of the counter *i*; TT_i – is the current value of the counter *i*; $Z_i=1$ – means counter is filled.

The analysis of changing logical signal (recognition of the front) is carried out by logical elements called signal front definer. Front definer differ by the character of changing logical signal: rising front:

$$\left.\begin{array}{l} Q_i=\begin{cases}1 & if\ (X_i(k-1)=0)\wedge(X_i(k)=1)\\ 0 & else;\end{cases}\\ Y_i=X_i(k);\\ q_i=\overline{Q_i},\\ where\ \ i=1,2,\ldots,n\end{array}\right\}, \qquad (6)$$

where n – is the number of definer in «the packet»; $X_i(k)$ – is the value of the signal being analyzed in the current clock period *kT*; $X_i(k-1)$ – is the value of the signal being analyzed in the previous clock period $(k-1)T$; Y_i – is the memory of the signal being analyzed; q_i – is absence of rising front, falling front:

$$\left.\begin{array}{l} Q_i=\begin{cases}1 & if\ (X_i(k-1)=1)\wedge(X_i(k)=0)\\ 0 & else;\end{cases}\\ Y_i=X_i(k)\\ q_i=\overline{Q_i},\\ where\ i=1,2,\ldots,n\end{array}\right\}, \qquad (7)$$

where n – is the number of definer in «the packet»; $X_i(k)$ – is the value of the signal being analyzed in the current clock period kT; $X_i(k-1)$ – is the value of the signal being analyzed in the previous clock period $(k-1)T$; Y_i – is the memory of the signal being analysed; q_i – is absence of falling front, (either) front:

$$\left.\begin{array}{l} Q_i=\begin{cases}1 & if\ X_i(k-1)\neq X_i(k)\\ 0 & else\end{cases}\\ Y_i=X_i(k)\\ q_i=\overline{Q_i}\\ where\ i=1,2,\ldots,n\end{array}\right\} \qquad (8).$$

where n – is the number of definer in «the packet»; $X_i(k)$ – is the value of the signal being analyzed in the current clock period *kT*; $X_i(k-1)$ – is the value of the signal being analyzed in the previous clock period $(k-1)T$; Y_i – is the memory of the signal being analyzed; q_i – is absence of front.

The signals having close inertance, that is having the same rate of changing are also included into «the packet» of logical signals. If the vector of logical signals is to contain signals of different inertance, then the time required to process the scheme containing the block of this module is defined by the signal of the highest rate of values. The definers of the front are also given in a «packet» form. At the time of changing signals at «packet» outputs corresponding logical values of signals appear during the time equal to one period of scheme processing if the signal change rate at the corresponding inputs isn't greater than that one at starting the control scheme.

Processing logical signals as arguments of multi place functions can be performed by means of so called vote function in schemes of recognition and correction of mistakes when information is transformed:

$$Y = \begin{cases} 1 & if \ \sum(X_1, X_2, \ldots, X_n) \geq m \\ 0 & else \end{cases}, \qquad (9)$$

where n – is the number of logical variables, m – quorum of vote system.

The dependence (9) defines the vote function conjunctively (majoritarian element) for any combination of logical signals from n by m. When $n = m$ the expression (9) corresponds to multi place conjunction, when $m = 1$ the expression (9) corresponds to multi place disjunction, when $n = m = 1$ the expression (9) corresponds to the function of repeat.

There are many different code converters. At designing sequential schemes, end automats, digital computers coders and decoders (these modules are also absent as standard ones according to IEC 61131-3) are especially important. The number of states of end automat corresponds to the number of inputs of coder and the number of outputs of a decoder [10, 14]. The function of a decoder is to transform any number of notation system X with base B from N positions, in the vector of logical signals having a dimension $m = B^N$. The signal at the output of the decoder Y_j with number $j = X$ is always equal to logical 1, $1 \leq j \leq m$, and values of the signals at other outputs Y_j with numbers $j \neq X$ being always equal to logical 0:

$$Y_j = \begin{cases} 1 & if \ j = X \\ 0 & for \ all \ j \neq X, \end{cases} \qquad (10)$$

where $j = 1, 2, \ldots, m, \quad 1 \leq X \leq m$.

Coder performs the operation being opposite that of decoder, that is it transforms the logical input vector $X_1, X_2, \ldots, X_m$ of dimension m into number j belonging to that input X_j which has a logical value equal 0. All inputs of a decoder but that one, which shows the current number of object state, must have zero values of signals:

$$Y = \begin{cases} 1 & if \ n \neq m \\ 0 & else, \end{cases} \qquad (11)$$

$$m = \begin{cases} i & for \ X_i = 1 \ and \ all \ X_j = 0, \ j \neq i \\ n & else, \end{cases} \qquad (12)$$

When being put in discrete signals are grouped in bytes, like real numbers, which later are used as logical signals. To get separate signals, not grouped into bytes the module of transforming sequence of real numbers into digit-by-digit position code, where each digit takes one byte is used.

When discrete signals are put out from outputs of combinatorial or sequential schemes it is necessary to transform the vector of logical signals into signals packed into bytes. For this purpose the module of transforming the vector of logical signals into a sequence of real numbers is used.

If signals oh the vector $X_1, X_2, \ldots, X_n$ can take arbitrary logical values, then the inputs with logical 1's can be «marked», that is can be indexed by dependence (indication register):

$$\left.\begin{array}{l} q = \sum(X_1, X_2, \ldots, X_n) \\ Y_j = i \cdot X_i \quad for \ all \ X_i \neq 0, \end{array}\right\} \qquad (13)$$

where q – is the number of inputs with value of the signal equal to 1, $X_i, j = 1, 2, \ldots, q$

Indication register is a function of coding with variable number of outputs. This module can be used for example for numbering inputs having value of signals equal to 1 (for example, numbering performing mechanisms whose state deviates from the norm). If values of signals at all n inputs are equal to 1 then the dependence will give the following values of outputs: $q = n, Y_1 = 1, Y_2 = 2, \ldots, Y_n = n$. If values at all inputs are 0, then $q = 0$, and outputs Y_j don't exist.

All variable connections in control system are performed by means of commutators. There are commutators of input signals (multiplexers or reading commutators), commutators of output signals (demultiplexers or distributors or writing commutators) and matrix commutators (multiplexers-demultiplexers or read/write commutators) [14]:

$$Y_i = (AC + EN + \{NV - 1\}NE + i), \quad m = 1 \qquad (14)$$

$$(AC + EN + \{NV - 1\}NE + i) = X_i, \quad m = 0 \qquad (15)$$

where i – 1, 2,…, NE; $m = 1$ reading; $m = 0$ writing; m – instruction to read/write; AC – the address of the signal; EN – number of the element; NV – number of the vector; NE – a number of elements. The function of a multiplexer we'll get by replacing $m = 1$ in the expression (14), and the function of a de multiplexer by replacing $m = 0$ in the expression (15). Each of the values being switched $X_1, X_2, \ldots, X_{NE}, Y_1, Y_2, \ldots, Y_{NE}$ may be a separate value of the signal (scalar) or a vector of values of a signal of a definite dimension and kind.

The sequence of low frequency impulses (getting signals of frequency 0,5 Hz; 1 Hz; 2 Hz;… for the synthesis of control schemes and warning) can be got by means of signal generators.

$$\left.\begin{array}{l} C(k)=\overline{C}(k-1); \\ Y_i=\begin{cases} C(k) & at\ X_i=1,\ i=1,2,\ldots,KK, \\ 0 & else, \end{cases} \end{array}\right\} \quad (16)$$

where *kk* – is the number of periodic pulse channels, *C(k)* – is the state of outputs at the moment *kT*; *C(k-1)* – is the state of outputs at the moment *(k-1)T;* X_i – is the state of pulses at input start/stop generation; Y_i – is the state of pulses at outputs.

Thus, discrete automatic devices can be implemented by programming according to formulae (1)-(16)

III CONCLUSION

Program implementation of main discrete automatic devices in complex SDSDC provides its universality at designing digital ACS. Such approach to the creation of industrial control systems allows to expend the set of standardized groups of discrete automatic modules without any essential expenses in comparison with the functional requirements of IEC 61131-3. The complex SDSDC as a design tool of digital ACS, is oriented not on professional programmers, but on engineers i.e. specialists in automation and remote control. The offered approach to the program implementation of discrete automatic devices allows to use the technique accepted at developing hardware systems. For example, the technique of developing and mounting control and regulation systems on the basis of pneumatics, hydraulics, electrical mechanics and discrete automatic devices based on chips of different scale integration.

IV REFERENCES

[1] Айзерман М.А., Гусев Л.А., Розоноэр Л.И. Логика. Автоматы. Алгоритмы.- М.: Государственное издательство физико-математической литературы, 1963, с. 556.

[2] Антонью А. Цифровые фильтры: анализ и проектирование.- М.: Радио и связь, 1983, с. 320.

[3] Баранов С.И. Синтез микропрограммных автоматов.- М.: Энергия, 1974, с. 216.

[4] Виши Г. Стандартные программы вычислительных машин для обработки данных. - М.: ВЦП, 1973, перевод № Ц-16684, с. 45.

[5] Гаврилов М.А., Девятков В.В., Пупырев Е.И. Логическое проектирование дискретных автоматов.- М.: Наука, 1977, с. 252.

[6] Горбатов В.А., Кафаров В.В., Павлов П.Г. Логическое управление технологическими процессами.- М.: Энергия, 1978, с. 272.

[7] Горбатов В.А., Крылов А.В., Федоров Н.В. САПР систем логического управления. М.: Энергоатомиздат, 1988, с. 230.

[8] Коневцов В.А. САПР цифровых САУ. Концепция: Монография. Издание третье, дополненное и исправленное: Псков: Псковский государственный университет, 2013. – с. 317.

[9] Коневцов В.А., Казаченко А.П., Литвинова Л.М., Бунин А.Б. Модифицированные средства цифрового управления.- М.: Информприбор, Каталог Государственной Системы Приборов СССР, 1987, том 4, вып. 10, 11, 12, с. 112.

[10] Лехин С.Н. Схемотехника ЭВМ.– СПб.: БХВ – Петербург, 2010, 661с.

[11] Микросхемы интегральные. Серии К1500…КР1531. Справочник.- С-Петербург, Издательство РНИИ «Электронстандарт», 1993, с.130.

[12] John Karl-Heinz, Tiegelkamp Michael SPS–Programmierung mit IEC 61131-3 .- Springer – Verlag Berlin Heidelberg 4. Auflage, 2009, s.402.

[13] Konevtsov V.A., Verteshev S.M., PoletaevI.A. Eigenschaften von Complex SDSDC // European Science and Technology: 7th International scientific conference. Germany, Munich 2014, Vol.I, p. 493-497.

[14] Seifart M. Digitale Schaltungen.- Berlin, VEB Verlag Technik, 1986, s. 560.

[15] Seitz M. Speicherprogrammierbare Steuerungen für die Fabrik- und Prozessautomation.- Fachbuchverlag Leipzig im Carl Hanser Verlag, 3. Auflage, 2012, s. 277.

[16] Speicherprogrammiebare Steuerungen. Teil 3: Programmiersprachen (IEC 61131-3:2003). Deutsche Fassung EN 61131-3:2003.

[17] Tröster F. Steuerungs- und Regelungstechnik für Ingenieure.- Oldenburg Verlag München, 3. Auflage, s. 562.

[18] Verteshev S.M., Konevtsov V.A., Poletaev I.A. Methods of Software Developing of Complex SDSDC // European Science and Technology: 4th International scientific conference. Germany, Munich 2013, p. 377-380.

[19] Verteshev S.M., Konevtsov V.A., Poletaev I.A. Softwaremittel der Projektierung von Systemen der digitalen Steuerung // European Science and Technology: 5th International scientific conference. Germany, Munich 2013, Vol. I, p. 501-504.

[20] Wellenreuter G., Zastrow D. Automatisierung mit SPS. Theorie und Praxis.- Vieweg + Teubner, 5. Auflage, 2011, s.870.