НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР «АРГО»®

Регистратор МУР-1001.2RC.
Описание интерпретатора ПАС.
v.11.08

Иваново 2011 г.

Содержание

1. Описание инструкций ПАС

Программа анализа событий подготавливается на языке анализа событий (ЯАС), сходном по синтаксису с языками технологического программирования. Программа в памяти регистратора представляет собой последовательность пятибайтных инструкций. 1-й байт инструкции – код операции, следующие 4 байта определяют используемый в инструкции операнд. Операндами инструкций могут быть:

- текущие и предыдущие (зафиксированные при предыдущем выполнении программы) состояния событий. Обозначаются соответственно ~EC.N и ~EP.N, где N – номер события (N = 0..255);

- текущие и предыдущие значения флагов определенности событий: соответственно ~DC.N и ~DP.N, где N – номер события (0..255);

- битовые переменные – меркеры ~M.N (N = 0..255) – используются для хранения промежуточных результатов логических операций;

- переменные с плавающей точкой – регистры ~R.N (N = 0..255) – используются для хранения результатов арифметических операций;

- флаги счетчиков - ~CT.N (N = 0..255). Счетчики используются для задания паузы при анализе логического выражения, например, для исключения дребезга сигналов от датчиков;

- флаги активности баз - ~BF.N (N = 0..7). Программа использует 2 образа флагов активности баз. 1-й образ представляет собой копию флагов активности баз, сформированных на оп. 1 (см. п. 2); используется в операциях, в которых флаг активности баз является источником. 2-й образ в начале выполнения программы обнуляется, служит для операций, использующих флаг активности баз в качестве приемника. При выполнении операций с флагом-приемником запись производится в оба образа. По окончании работы программы содержимое второго образа переписывается в флаги активности баз. Такой механизм дает возможность запустить процедуру формирования записи для какой-либо из баз по результатам работы программы анализа событий (запись будет сформирована при следующем выполнении процедуры формирования записей);

- параметры из текущей и предыдущей записей баз данных - ~FC.B.O.L и ~FP.B.O.L. Поле B задает номер базы (0..7), поле O – смещение параметра относительно начала записи, поле L – формат параметра (BCD, Unsigned Bin, Signed Bin, Float)- старшая тетрада и длина параметра в байтах – младшая тетрада. ~FC определяет параметр из текущей записи базы (т.е., самой последней по времени записи), ~FP – из предыдущей (предпоследней записи);

- байты из текущей и предыдущей записей баз данных - ~BC.B.O.M и ~BP.B.O.M. Поле B – номер записи (0..7), поле O – смещение байта относительно начала записи, поле M – маска. Значение байта, используемое программой, определяется как Byte AND M, где Byte – байт со смещением O из указанной записи базы B;

- переменные из ОЗУ, Flash, EEPROM - ~RF.Addr.L, ~FF.Addr.L, ~EF.Addr.L. Addr – адрес переменной, L – формат переменной и ее длина;

- байты из ОЗУ, Flash, EEPROM - ~RB.Addr.M, ~FB.Addr.M, ~EB.Addr.M. Addr – адрес переменной, M – маска. Значение байта, используемое программой, определяется как Byte AND M, где Byte – байт из заданного адреса банка памяти;

- константы ~C.Value, где Value – значение с плавающей точкой,

- указатели текущей и предыдущей записей - ~PC.B, ~PP.B. Поле B (0..7) – номер базы.

Внутреннее представление операндов различных типов приведено в табл. 1.1.

Табл. 1.1.

Внутреннее представление операндов программы анализа событий

Тип операнда	**Обозначение**	**Внутреннее представление (байты 2..5 инструкции) (HEX)**
Текущее состояние события	~EC.N	00 N XX XX
Предыдущее состояние события	~EP.N	08 N XX XX
Текущее значение флага определенности	~DC.N	10 N XX XX
Предыдущее значение флага определенности	~DP.N	18 N XX XX
Битовая переменная – меркер	~M.N	20 N XX XX
Переменная с плавающей точкой - регистр	~R.N	28 N XX XX
Флаг счетчика	~CT.N	30 N XX XX
Флаг активности базы	~BF.B	1-й байт – 0011 1bbb, bbb – номер базы. Байты 2..4 не используются
Параметр из текущей записи	~FC.B.O.L	1-й байт – 0100 0bbb, bbb – номер базы B; 2-й байт – младший байт смещения O, 3-й байт – старший байт смещения O, 4-й байт – старшая тетрада- формат параметра: 0 – BCD (двоично-десятичный код); 4 – Unsigned Bin (беззнаковое двоичное); 8 – Float BCD (число с плавающей точкой); C – Signed Bin (двоичное со знаком), 4-й бат – младшая тетрада – длина операнда.

Продолжение табл. 1.1.

Параметр из предыдущей записи	~FP.B.O.L	1-й байт – 0100 1bbb, bbb – номер базы B; 2-й байт – младший байт смещения O, 3-й байт – старший байт смещения O, 4-й байт – старшая тетрада- формат параметра: 0 – BCD (двоично-десятичный код); 4 – Unsigned Bin (беззнаковое двоичное); 8 – Float BCD (число с плавающей точкой); C – Signed Bin (двоичное со знаком), 4-й бат – младшая тетрада – длина операнда.
Байт из текущей записи	~BC.B.O.M	1-й байт – 0101 0bbb, bbb – номер базы B; 2-й байт - младший байт смещения O, 3-й байт – старший байт смещения O, 4-й байт – маска M.
Байт из предыдущей записи	~BP.B.O.M	1-й байт – 0101 1bbb, bbb – номер базы B; 2-й байт - младший байт смещения O, 3-й байт – старший байт смещения O, 4-й байт – маска M.
Переменная из ОЗУ	~RF.Addr.L	1-й байт – 0110 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – старшая тетрада- формат параметра: 0 – BCD (двоично-десятичный код); 4 – Unsigned Bin (беззнаковое двоичное); 8 – Float BCD (число с плавающей точкой); C – Signed Bin (двоичное со знаком), 4-й бат – младшая тетрада – длина операнда. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.
Байт из ОЗУ	~RB.Addr.M	1-й байт – 0111 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – маска M. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.
Переменная из Flash	~FF.Addr.L	1-й байт – 1000 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – старшая тетрада- формат параметра: 0 – BCD (двоично-десятичный код); 4 – Unsigned Bin (беззнаковое двоичное); 8 – Float BCD (число с плавающей точкой); C – Signed Bin (двоичное со знаком), 4-й бат – младшая тетрада – длина операнда. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.
Байт из Flash	~FB.Addr.M	1-й байт – 1001 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – маска M. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.
Переменная из EEPROM	~EF.Addr.L	1-й байт – 1010 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – старшая тетрада- формат параметра: 0 – BCD (двоично-десятичный код); 4 – Unsigned Bin (беззнаковое двоичное); 8 – Float BCD (число с плавающей точкой); C – Signed Bin (двоичное со знаком), 4-й бат – младшая тетрада – длина операнда. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.
Байт из EEPROM	~EB.Addr.M	1-й байт – 1011 aaaa, aaaa – младшая тетрада адреса Addr, 2-й байт – младший байт выражения Addr DIV 10H, 3-й байт - старший байт выражения Addr DIV 10H, 4-й байт – маска M. Если заданный в инструкции адрес больше 7FFFFh, то адрес вычисляется как сумма младших 19 разрядов, заданных в инструкции, и значения регистра ~R.0.

Продолжение табл. 1.1.

Константа	~C.Value	1-й байт – 1100 abcd, a – бит 5 характеристики, b – бит 4 характеристики, c – знак характеристики (1 -минус), d – знак мантиссы (1 – минус), 2-й байт – цифры 4..5 мантиссы (BCD-код), 3-й байт – цифры 2..3 мантиссы (BCD-код), 4-й байт – старшая тетрада – 1-я цифра мантиссы, 4-й байт – младшая тетрада – биты 0..3 характеристики. Константа представляется в нормализованной форме, т.е., модуль ненулевой константы меньше 1 и не меньше 0.1. Мантисса представляется числом с 5 значащими цифрами. Диапазон характеристик мантиссы: $10^{-63}..10^{+63}$. Если байт 4 = 0 – константа равна 0.
Указатель на текущую запись	~PC.B	1101 0bbb. bbb – номер базы.
Указатель на предыдущую запись	~PP.B	1101 1bbb. bbb – номер базы.
Адрес перехода		Адрес перехода представляет собой двухбайтное число, равное номеру инструкции, на которую осуществляется переход. 1-й байт - младший байт адреса перехода, 2-й байт – старший байт адреса перехода.

Программа анализа событий использует следующие внутренние переменные:

- бит-результат логических операций (RLO). В большинстве логических операциях RLO используется в качестве операнда и результата операции;
- аккумулятор (ACC). Используется в качестве операнда и результата в арифметических операциях;
- счетчик инструкций (PC) – двухбайтное число, равное номеру выполняемой инструкции;
- битовый стек (BS). Предназначен для хранения промежуточных результатов логических операций. Глубина битового стека равна 8;
- стек инструкций (CS). Используется для хранения адресов возврата из подпрограмм. Глубина стека инструкций равна 8;
- статус программы.

Все инструкции можно разделить на следующие группы:

А. Битовые инструкции. Операндами битовых инструкций могут быть биты ~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, а также ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB и в качестве операндов-источников (но не приемников!) ~FC, ~FP, ~BC, ~BP, ~C. Если операнд-источник битовой инструкции типа ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP или ~C, то при выполнении операции он преобразуется к битовому типу. Если float-представление операнда равно 0, в битовой инструкции в качестве операнда-источника используется бит 0; если float-представление операнда не равно 0, то бит = 1. При использовании в битовых операциях ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB в качестве приемников битовый результат = 0 преобразуется к float-представлению 0, результат =1 – к представлению 1.0.

Список битовых инструкций приведен в табл. 1.2.

Табл. 1.2.

Список битовых инструкций

Код (HEX)	**Мнемоническое обозначение**	**Описание**	**Допустимые операнды**	**Модификация** (+ - модифицируется, - - не изменяется, 0 – сбрасывается, 1- устанавливается)		**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
				RLO	Операнд	
01	**L** Operand	Загрузить в RLO значение операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~M.10=0 **L ~M.10** RLO=0, ~M.10=0
02	**LN** Operand	Загрузить в RLO инверсное значение операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~EP.1=1 **LN ~EP.1** RLO=0, ~EP.1=1
03	= Operand	Присвоить операнду значение RLO.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	-	+	RLO=1, ~M.3=0 **= ~M.3** RLO=1, ~M.3=1
04	**=N** Operand	Присвоить операнду инверсное значение RLO.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	-	+	RLO=0, ~M.3=0 **=N ~M.3** RLO=0, ~M.3=1
05	**A** Operand	Логическое «И» RLO и операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=0, ~DP.4=1 **A ~DP.4** RLO=1, ~DP.4=1

Продолжение табл. 1.2.

06	**AN** Operand	Логическое «И» RLO и инверсного значения операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~CT.2=0 **AN ~CT.2** RLO=1, ~CT.2=0
07	**O** Operand	Логическое «ИЛИ» RLO и операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=0, ~R.4=56.7 **O ~R.4** RLO=1, ~R.4=56.7
08	**ON** Operand	Логическое «ИЛИ» RLO и инверсного значения операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~M.4=1 **ON ~M.4** RLO=1, ~M.4=1
09	**X** Operand	Логическое «Исключающее ИЛИ» RLO и операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~EP.2=1 **X ~EP.2** RLO=0, ~EP.2=1
0A	**XN** Operand	Логическое «Исключающее ИЛИ» RLO и инверсного значения операнда.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	RLO=1, ~EP.2=1 **XN ~EP.2** RLO=1, ~EP.2=1
0B	**(L** Operand	Записать значение RLO в битовый стек, загрузить в RLO значение операнда. Байт битового стека сдвигается влево.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=0, RLO=1, ~M.4=0 **(L ~M.4** BS=1, RLO=0, ~M.4=0
0C	**(LN** Operand	Записать значение RLO в битовый стек, загрузить в RLO инверсное значение операнда. Байт битового стека сдвигается влево.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=05h, RLO=1, ~M.4=0 **(LN ~M.4** BS=0bh, RLO=1, ~M.4=0
0D	**A)**	Логическое «И» RLO и бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=11h, RLO=0 **A)** BS=08h, RLO=0
0E	**AN)**	Логическое «И» RLO и инверсного значения бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=11h, RLO=0 **AN)** BS=08h, RLO=0
0F	**O)**	Логическое «ИЛИ» RLO и бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=05h, RLO=0 **O)** BS=02h, RLO=1
10	**ON)**	Логическое «ИЛИ» RLO и инверсного значения бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=05h, RLO=0 **ON)** BS=02h, RLO=0
11	**X)**	Логическое «Исключающее ИЛИ» RLO и бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=2h, RLO=0 **X)** BS=1, RLO=0
12	**XN)**	Логическое «Исключающее ИЛИ» RLO и инверсного значения бита из стека. Байт битового стека сдвигается вправо.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	+	-	BS=2h, RLO=0 **XN)** BS=1, RLO=1
13	**SR**	Установить RLO. RLO = 1.	Инструкция без операнда	1	-	RLO=0 **SR** RLO=1
14	**RR**	Сбросить RLO. RLO = 0.	Инструкция без операнда	0	-	RLO=1 **RR** RLO=0
15	**CR**	Инвертировать RLO.	Инструкция без операнда	+	-	RLO=0 **CR** RLO=1
16	**S** Operand	Установить бит Operand. Operand = 1.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	-	1	RLO=1, ~R.3=0.12 **S ~R.3** RLO=1, ~R3.0=1.0
17	**R** Operand	Сбросить бит Operand. Operand = 0.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	-	0	RLO=1, ~M.3=1 **R ~M.3** RLO=1, ~M3.0=0
18	**C** Operand	Инвертировать бит Operand.	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	-	+	RLO=1, ~R.3=1.12 **C ~R.3** RLO=1, ~R.3=0.0

В. Арифметические инструкции. Одним из операндов большинства арифметических операций является аккумулятор ACC. Другим операндом могут быть ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, а также биты ~EC, ~EP , ~DC,

~DP, ~M, ~CT, ~BF, и в качестве операндов-источников (но не приемников!) ~FC, ~FP, ~BC, ~BP, ~C. Если операнд-источник арифметической инструкции типа ~EC, ~EP , ~DC, ~DP, ~M, ~CT или ~BF, то при выполнении операции он преобразуется к float-представлению: нулевое значение бита соответствует float-числу 0.0, единичное значение бита – float-числу 1.0. Если float-представление результата операции равно 0.0 – бит-приемник ~EC, ~EP , ~DC, ~DP, ~M, ~CT или ~BF сбрасывается, иначе- бит-приемник устанавливается в 1. Арифметические инструкции не влияют на бит RLO.

Список арифметических инструкций приведен в табл. 1.3.

Табл. 1.3.

Список арифметических инструкций.

Код (HEX)	**Мнемоническое обозначение**	**Описание**	**Допустимые операнды**	**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
19	**LF** Operand	Загрузить в ACC значение операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.2, ~R.4=12.35 **LF ~R.4** ACC=12.35, ~R.4=12.35
1a	**LFN** Operand	Загрузить в ACC отрицательное значение операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.2, ~R.4=12.35 **LFN ~R.4** ACC=-12.35, ~R.4=12.35
1b	**=F** Operand	Присвоить операнду значение ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	ACC=3.5, ~R.1=10.0 **=F ~R.1** ACC=3.5, ~R.1=3.5
1c	**=FN** Operand	Присвоить операнду отрицательное значение ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	ACC=3.5, ~R.1=10.0 **=FN ~R.1** ACC=3.5, ~R.1=-3.5
1d	+ Operand	Сложить ACC и операнд. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=10.0 **+ ~R.1** ACC=13.5, ~R.1=10.0
1e	- Operand	Вычесть из ACC значение операнда. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=10.0 **- ~R.1** ACC=-6.5, ~R.1=10.0
1f	* Operand	Умножить ACC на операнд. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=10.0 *** ~R.1** ACC=35.0, ~R.1=10.0
20	/ Operand	Разделить ACC на операнд. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=10.0 **/ ~R.1** ACC=0.35, ~R.1=10.0
21	**CLA**	Обнулить ACC	Инструкция без операнда	ACC=2.7 **CLA** ACC=0.0
22	**CLF** Operand	Обнулить операнд	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	~R.5=2.7 **CLF ~R.5** ~R.5=0.0
23	**NA**	Изменить знак ACC	Инструкция без операнда	ACC=4.7 **NA** ACC=-4.7
24	**NF** Operand	Изменить знак операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB	~R.1=7.3 **NF ~R.1** ~R.1=-7.3 ~EC.2=0 **NF ~EC.2** Бит ~EC.2 преобразуется к типу float (0.0), изменяется знак результата (-0.0=0.0), затем выполняется преобразование результата к битовому типу. Поэтому после инструкции ~EC.2=0
41	**ABS**	Взять модуль ACC	Инструкция без операнда	ACC=-167.3 **ABS** ACC= 167.3
42	**INT**	Взять целую часть ACC.	Инструкция без операнда	ACC=102.3 **INT** ACC=102
43	**FRAC**	Взять дробную часть ACC	Инструкция без операнда	ACC=102.3 **FRC** ACC=0.3
44	**DIV** Operand	Разделить целую часть ACC на целую часть операнда нацело. Результат в ACC	~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=12.6 ~R.8=2.8 **DIV ~R.8** ACC=6

Продолжение табл. 1.3

45	**MOD** Operand	Взять остаток от целочисленного деления целой части модуля ACC целую часть операнда. Результат в ACC	~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=12.6 ~R.1=5.99 **MOD ~R.1** ACC=2

С. Байтовые инструкции. В байтовых операциях участвуют 2 операнда:

1-й операнд – 4 байта двоичного представления целой части аккумулятора ACC;

2-й операнд – 4 байта двоичного представления целой части указанного в инструкции операнда.

Если операнд инструкции типа ~EC, ~EP , ~DC, ~DP, ~M, ~CT или ~BF, то при выполнении операции он преобразуется к float-представлению: нулевое значение бита соответствует float-числу 0.0, единичное значение бита – float-числу 1.0. ACC и float-представление операнда при выполнении байтовых инструкций преобразуются к целым четырехбайтным числам со знаком. Если числа (ACC или операнд) больше 2147483647 (7FFFFFFFh), то значение числа принимается равным 2147483647, если ACC или операнд меньше -21474836483, то -2147483647. Если значение ACC или операнда по модулю меньше 1, то в инструкции участвует нулевое значение.

Операции сдвига при нулевом значении операнда не модифицируют ACC. Если двоичное представление операнда больше или равно 32, или операнд отрицательный, то результатом операций сдвигов влево или вправо будет 0.

Результат байтовых инструкций в ACC, 2-й операнд не модифицируется. На бит RLO байтовые инструкции не влияют.

Список байтовых инструкций приведен в табл. 1.4.

Табл. 1.4.

Список байтовых инструкций

Код (HEX)	**Мнемоническое обозначение**	**Описание**	**Допустимые операнды**	**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
25	**AB** Operand	Поразрядное «Логическое И» ACC и операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=7.2, ~R.4=12.35 **AB ~R.4** ACC=4.0, ~R.4=12.35
26	**ABN** Operand	Поразрядное «Логическое И» ACC и отрицательного значения операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.2, ~R.4=1 **ABN ~R.4** ACC=3.0, ~R.4=12.35
27	**OB** Operand	Поразрядное «Логическое ИЛИ» ACC и операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=10.5 **OB ~R.1** ACC=11.0, ~R.1=10.5
28	**OBN** Operand	Поразрядное «Логическое ИЛИ» ACC и отрицательного значения операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=-1.42 **=OBN ~R.1** ACC=3.0, ~R.1=-1.42
29	**XB** Operand	Поразрядное «Логическое Исключающее ИЛИ» ACC и операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=1.5, ~R.1=2.9 **XB ~R.1** ACC=3.0, ~R.1=2.9
2a	**XBN** Operand	Поразрядное «Логическое Исключающее ИЛИ» ACC и отрицательного значения операнда	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=3.5, ~R.1=-1.0 **XBN ~R.1** ACC=2.0, ~R.1=-1.0
46	<< Operand	Сдвиг влево целой части ACC на целую часть операнда разрядов. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=5.9 ~EC.0=1 **<< ~EC.1** ACC=10
47	>> Operand	Сдвиг вправо целой части ACC на целую часть операнда разрядов. Результат в ACC	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	ACC=5.9 ~M.10=1 **<< ~M.10** ACC=2

D. Инструкции сравнения. В инструкциях сравнения значение аккумулятора ACC сравнивается со значением второго операнда, как числа с плавающей точкой. Бит RLO устанавливается в 1, если логический результат сравнения – «истина», иначе RLO сбрасывается. ACC и второй операнд не модифицируются. Если второй операнд типа ~EC, ~EP , ~DC, ~DP, ~M, ~CT или ~BF, то при выполнении операции он преобразуется к float-представлению: нулевое значение бита соответствует float-числу 0.0, единичное значение бита – float-числу 1.0. Список инструкций сравнения приведен в табл. 1.5.

Табл. 1.5.

Список инструкций сравнения

Код (HEX)	**Мнемоническое обозначение**	**Эквивалентная математическая запись**	**Допустимые операнды**	**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
2b	**LT** Operand	ACC < Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=7.2, ~R.4=12.35 **LT ~R.4** RLO=1, ACC=7.2, ~R.4=12.35
2c	**LE** Operand	ACC ≤ Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=3.2, ~R.4=1 **LE ~R.4** RLO=0, ACC=3.2, ~R.4=1
2d	**EQ** Operand	ACC = Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=3.5, ~R.1=10.5 **EQ ~R.1** RLO=0, ACC=3.5, ~R.1=10.5
2e	**NE** Operand	ACC ≠ Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=3.5, ~R.1=-1.42 **NE ~R.1** RLO=1, ACC=3.5, ~R.1=-1.42
2f	**GT** Operand	ACC > Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=1.5, ~R.1=2.9 **GT ~R.1** RLO=0, ACC=1.5, ~R.1=2.9
30	**GE** Operand	ACC ≥ Operand	~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~BF, ~R, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP, ~C	RLO=1, ACC=3.5, ~R.1=-1.0 **GE ~R.1** RLO=1,ACC=3.5, ~R.1=-1.0

E. Инструкции для работы со счетчиками. В эту группу входят всего 2 команды: инструкция загрузки счетчика и инструкция сброса счетчика. Счетчик с точки зрения программиста представляет собой запись, включающую 2 поля:

- флаг счетчика ~CT- битовый параметр, который может выступать операндом логических, байтовых и арифметических операции;

- 16-разрядный регистр счетчика, доступ к которому осуществляется только по записи в инструкциях для работы со счетчиком.

После включения питания регистратора процедура инициализации ПАС сбрасывает флаги всех используемых в программе счетчиков и записывает в регистры счетчиков значение 0ffffh (65535). Операция загрузки счетчика **LC ~CT.N** (N – номер счетчика, N< 256) работает следующим образом:

- если бит RLO=1 и значение регистра счетчика не равно 65535, то значение регистра счетчика декрементируется. Если после декремента в регистре счетчика 0, то устанавливается флаг счетчика, иначе флаг счетчика сбрасывается;

- иначе, если:

а) бит RLO=1 и значение регистра счетчика равно 65535 или

б) бит RLO=0,

то в регистр счетчика загружается значение младших двух байт целой части ACC (ACC преобразуется к 4-хбайтному целому числу со знаком, в счетчик загружаются 2 младших байта. Корректные значения после преобразования будут в том случае, если целая часть аккумулятора X в диапазоне: -2147483648 <= X <= 2147483647.). Если в регистре счетчика 0 – флаг счетчика устанавливается, иначе флаг счетчика сбрасывается.

Вторая инструкция для работы со счетчиком – сброс счетчика **RC ~CT.N** (N – номер счетчика) сбрасывает флаг счетчика и записывает в регистр счетчика 0ffffh – т.е., аналогична процедуре инициализации ПАС.

Бит RLO и аккумулятор ACC при выполнении инструкций со счетчиком не модифицируются.

Счетчики используются в ПАС для организации временных пауз, например, при обработке сигналов с датчиков дискретного ввода для исключения дребезга. Величина временной паузы равна произведению периода вызова ПАС на загружаемое в регистр счетчика значение.

Список инструкций для работы со счетчиком приведен в табл. 1.6..

Табл. 1.6.

Список инструкций для работы со счетчиками

Код (HEX)	**Мнемоническое обозначение**	**Описание**	**Допустимые операнды**	**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
31	**LC** Operand	Загрузить в регистр счетчика значение ACC	~CT	RLO=1, регистр ~CT.2=1, флаг ~CT.2=0 **LC ~CT.2** RLO=1, регистр ~CT.2=0, флаг ~CT.2=1
32	**RC** Operand	Сбросить счетчик	~CT	RLO=1, регистр ~CT.2=1, флаг ~CT.2=0 **RC ~CT.2** RLO=1, регистр ~CT.2=0ffffh, флаг ~CT.2=0

F. Инструкции переходов. Делятся на подгруппу инструкций условных переходов (коды 33h.. 38h) и подгруппу инструкций безусловных переходов (коды 39h..3bh). Для инструкций первой подгруппы если условие перехода не выполняется, то значение счетчика инструкций инкрементируется, т.е., будет выполняться следующая за условным переходом инструкция. Операндами инструкций переходов являются адреса переходов, равные номеру инструкции, к которой осуществляется переход. Инструкции переходов не модифицируют RLO и ACC.

Список инструкций переходов приведен в табл. 1.7.

Табл. 1.7.

Список инструкций переходов

Код (HEX)	**Мнемоническое обозначение**	**Описание**	**Пример инструкции** (состояние до выполнения инструкции, инструкция, состояние после выполнения инструкции)
33	**JR** Address	Переход, если RLO=1	RLO=1, PC=0010h **JR 0023h** RLO=1, PC=0023h
34	**JNR** Address	Переход, если RLO=0	RLO=1, PC=0010h **JNR 0023h** RLO=1, PC=0011h
35	**JP** Address	Переход, если ACC > 0	ACC=0.0, PC=0005h **JP 0012h** ACC=0.0, PC=0006h
36	**JM** Address	Переход, если ACC < 0	ACC=1.0, PC=0005h **JM 0012h** ACC=1.0, PC=0006h
37	**JZ** Address	Переход, если ACC=0.0	ACC=0.0, PC=0005h **JZ 0012h** ACC=0.0, PC=0012h
38	**JNZ** Address	Переход, если ACC ≠ 0.0	ACC=0.0, PC=0005h **JNZ 0012h** ACC=0.0, PC=0006h
39	**JMP** Address	Безусловный переход	PC=0008h **JMP 0012h** PC=0012h
3a	**CALL** Address	Переход к подпрограмме	PC=0003h, указатель стека инструкций = 1 **CALL 0010h** PC=0010h, указатель стека инструкций =2, в стек инструкций записывается 0004h (адрес следующей за вызовом подпрограммы инструкцией)
3b	**RET**	Возврат из подпрограммы	PC=0010h, указатель стека инструкций =2 **RET** PC считывается из стека инструкций, указатель стека инструкций декрементируется

G. Инструкции преобразования даты/времени.

CD Operand (код 3ch). По этой инструкции значение даты/времени записи базы, начиная с указанного поля, преобразуется в секунды, полученное значение загружается в ACC.Операндами инструкции могут быть только ~FC или ~FP, указывающие на одно из полей даты/времени записи базы данных. Преобразование выполняется с учетом формата поля **DateTime** указанной базы. Поля, старшие заданного в инструкции, при преобразовании игнорируются, отсутствующие младшие поля **DateTime** базы принимаются равными 0.

Например, запись из базы номер 3 включает 4 поля даты/времени, старшее поле – месяц. Первые 4 байта текущей записи следующие (BCD-код):

06 02 14 30 (т.е., 2 июня, 14 часов 30 минут).

После выполнения инструкции **CD ~FC.3.1.1** (ссылка на сутки) в ACC будет число 138600 (количество секунд от 00:00:00 1-го числа до 14:30:00 2-го числа – т.е., 24+14=38 часов 30 минут 00 секунд). Результатом **CD ~FC.3.3.1** (ссылка на минуты) будет ACC = 1800 (количество секунд от 00 минут до 30 минут этого же часа).

WD Operand (код 3fh). По этой инструкции в ACC возвращается значение дня недели для указанного операнда (0- воскресенье, 1- понедельник, .. 6- суббота). Операндами инструкции могут быть только ~FC или ~FP, указывающие на начало записи даты/времени. Преобразование выполняется с учетом формата поля **DateTime** указанной базы. Поля года, месяца, дня, не заданные в инструкции (старшие заданного), принимаются равными соответственно 0, 1 и 1.

Например, запись базы номер 3 включает 4 поля даты/времени, старшее поле – год. Первые 4 байта текущей записи следующие:

09 04 27 12 (27 апреля 2009 г., 12 часов). После выполнения инструкции **WD ~FC.0.0.1** в ACC будет значение 1 (понедельник), а после **WD ~FC.0.1.1** – ACC=4 (четверг, день недели на 27 апреля 2000 г.).

Инструкции преобразования даты/времени не модифицируют RLO.

H. Инструкции проверки записей. Эта группа также представлена единственной инструкцией. Код инструкции – 3dh, мнемоническое обозначение **CB Operand.** Операндом инструкции могут быть указатели записи ~PC или ~PP. По инструкции выполняется проверка контрольной суммы указанной записи. Результат возвращается в бите RLO: RLO=1 – контрольная сумма верная, запись корректна; RLO=0 – не совпадает контрольная сумма, данные в записи не достоверны.

I. Прочие инструкции. В этой группе оказались 5 инструкций, не вошедшие в вышеперечисленные группы.

N**OP** (код 00h) – операция, не выполняющая никаких действий. Операнд в этой инструкции отсутствует.

ML (код 3eh) – вызов процедуры добавления записи в журнал операций. В результате выполнения инструкции в журнал операций будет добавлена запись с полем **Code**, равным младшему байту целой части аккумулятора. Операнд в этой инструкции отсутствует.

MR Operand (код 40h) – вызов процедуры формирования записи в базе. Номер базы задается значением **Operand**. Операндом инструкции может быть только ~BF, заданный в инструкции номер базы должен соответствовать одному из 2 типов баз: периодической или периодической по изменениям. Формирования новой записи в базе может занимать значительное время, необходимое для работы с устройствами, параметры которых включены в заданную базу. Соответственно увеличивается время выполнения ПАС.

Wait Operand (код 48h) – задержка выполнения ПАС на Operand миллисекунд. Корректные значения операнда в диапазоне 0..3276750 мс. Заданное значение операнда округляется до ближайшего минимального значения, кратного 50 мс.

END (код 0ffh) – инструкция завершения работы ПАС. Операнд в этой инструкции отсутствует. После выполнения инструкции END работа ПАС завершается.

2. Процедура инициализации ПАС и обработка ошибок

Процедура инициализации ПАС выполняется при каждом включении питания регистратора. В начале инициализации ПАС проверяется корректность описания баз и блока управления программой. При некорректном описании код статуса программы устанавливается равным 7fh. Если описания корректны, анализируется байт разрешения работы ПАС. Если биты 4,7 байта разрешения работы ПАС установлены, а биты 5,6 сброшены – работа ПАС разрешена, код статуса обнуляется. Другие комбинации битов 4..7 байта разрешения работы ПАС запрещают работу ПАС, в код статуса в этом случае записывается 80h. При инициализации ПАС счетчик инструкций, битовый стек и аккумулятор обнуляются, сбрасываются флаги счетчиков, в регистры счетчиков записывается значение 0ffffh. Биты 0..3 байта разрешения работы ПАС управляют инициализацией внутренних переменных ПАС:

- бит 0 = 1 – сброс битов ~EP;
- бит 1 = 1 – сброс битов ~DP;
- бит 2 = 1 – сброс меркеров ~M;
- бит 3 = 1 – обнуление регистров ~R.

При нулевых значениях битов 0..3 байта разрешения работы ПАС инициализация соответствующей области внутренних переменных ПАС не выполняется.

Нулевое значение кода статуса разрешает автоматическое выполнение ПАС по активизации любой из событийных баз. Значение кода статуса равное 80h разрешает только пошаговое выполнение программы по интерфейсным командам (режим отладки программы). Другие значения кода статуса запрещают работу ПАС. При возникновении фатальных ошибок при работе ПАС в байт статуса записывается код ошибки, сбрасывается флаг разрешения работы ПАС, и работа ПАС прекращается. Если настройки регистратора предусматривают ведение журнала операций, то при появлении ошибки ПАС в журнал добавляется запись, содержащая код ошибки и номер инструкции, выполнение которой привело к фатальной ошибке. Продолжение работы ПАС возможно после выполнения инициализации ПАС и установки флага разрешения работы ПАС.

Коды фатальных ошибок приведены в табл. 2.1.

Табл. 2.1.

Коды фатальных ошибок ПАС

Код ошибки (HEX)	Описание
01	Неизвестная инструкция: 1-й байт инструкции не совпадает ни с одним кодом операции
02	Недопустимый (неизвестный) операнд: код операнда не соответствует коду операции или код операнда отличается от кодов определенных операндов
03	Переполнение битового стека: попытка записи в битовый стек более 8 битовых значений
04	Антипереполнение битового стека: попытка чтения из пустого битового стека
05	Переполнение стека инструкций: уровень вложенности подпрограмм более 8
06	Антипереполнение стека инструкций: попытка выполнить инструкцию **RET** без инструкции **CALL**
07	Деление на 0: делитель (Operand) в инструкции **/ Operand** равен 0
08	Некорректное значение счетчика инструкций PC: в результате выполнения инструкции счетчику PC присвоено значение, превышающее максимально допустимое

3. Инструментальное программное обеспечение ПАС

3.1. Состав и взаимодействие компонентов инструментального ПО

Специфика программ анализа событий заключается в использовании в качестве операндов инструкций полей баз данных регистратора. Структура баз данных, периодичность ведения баз, количество и тип подключенных к регистратору устройств, список включаемых в базу параметров в общем случае являются уникальными для каждого регистратора. С другой стороны, повторение полного цикла разработки ПАС для каждого отдельного регистратора нерационально, необходимо иметь возможность переноса фрагментов программ или программ в целом с одного регистратора на другой по возможности с минимальными затратами времени. Это требование определило состав инструментального ПО и определенный алгоритм разработки ПАС.

Инструментальное программное обеспечение представляет собой следующий набор процедур:

1. Процедура генерации структур баз.
2. Процедура генерации символических имен.
3. Процедура определения символических имен.
4. Компилятор.
5. Загрузчик.
6. Отладчик.
7. Декодировщик.

На первом этапе подготовки ПАС генерируется структура баз данных регистратора, т.е., строится список полей баз данных, включающий название параметра, смещение параметра относительно начала записи, формат и длину параметра. Структура баз данных записывается в файл, имя которого соответствует серийному номеру регистратора, а расширение - '.bas'. Подготовка файла со структурой базы данных производится программой CfgWin2RC.exe («Регистратор»/«Сервис» /«Создать файл описания баз»). Последующие этапы подготовки ПАС выполняются с помощью программы PAS.exe.

Исходный текст программы анализа событий подготавливается на языке, синтаксис которого описан в п. 1. В качестве операндов могут использоваться как абсолютные операнды, перечисленные в табл. 1.1 (например, ~C.-0.1E-5 – константа $0.1*10^{-5}$, ~C.3 – константа 3), так и символические имена операндов и меток. После подготовки исходного текста программы (файл с расширением '.src') выполняется процедура генерации символических имен, т.е., строится список используемых в программе имен операндов и меток, определяются номера инструкций, соответствующих меткам. Список символических имен записывается в файл, которому присваивается имя файла с исходным текстом программы и расширение '.de_'. При обнаружении ошибок (повторное определение меток, пропущены операнды инструкции и т.д.) файл со списком не генерируется, а обнаруженные ошибки записываются в файл '.err'.

Третий этап подготовки ПАС заключается в определении используемых в программе символических имен. В режиме диалога каждому имени из файла '.de_' ставится в соответствие абсолютный операнд. Для выбора параметров из баз данных используются записи '.bas'-файла. Описания символических имен включаются в файл '.de_', т.е., файл '.de_' при определении символических имен переписывается.

На четвертом этапе производится компиляция ПАС. Исходной информацией для компиляции служит файл '.src' с текстом программы на ЯАС и файл с описанием символических имен '.de_'. Результат компиляции- файл с листингом программы (расширение '.lst') и файл '.cod' с кодами загружаемой в регистратор программы. Обнаруженные при компиляции ошибки выделяются в файле листинга, при обнаружении ошибок файл '.cod' не генерируется.

Пятым этапом подготовки ПАС является загрузка скомпилированной программы (файл '.cod') в регистратор. На этом этапе определяется адрес области памяти для хранения программы, резервируется область ОЗУ для рабочих переменных, программа загружается в регистратор.

На шестом этапе выполняется отладка программы. Отладка может происходить в двух режимах:

- в режиме эмуляции, т.е., программа анализа событий выполняется на компьютере без использования регистратора. В режиме эмуляции поддерживается работа только с внутренними переменными программы, т.е., с ~EC, ~EP , ~DC, ~DP, ~M, ~CT, ~R и ~C. Результаты операций с операндами ~BF, ~RF, ~RB, ~FF, ~FB, ~EF, ~EB, ~FC, ~FP, ~BC, ~BP не определены и должны корректироваться с клавиатуры;
- отладка с использованием регистратора. В этом режиме компьютер дает регистратору интерфейсную команду на выполнение очередной инструкции и затем считывает из регистратора результат выполнения инструкции.

В обоих режимах предусмотрено пошаговое выполнение программы, задание точки останова, возможность редактирования переменных. При отладке используется листинг.

Подготовка ПАС может быть повторена с любого этапа. При переносе программы на другой регистратор исходный файл не модифицируется, адаптация ПАС сводится к генерации структуры базы, и переопределению символических имен, компиляции и загрузке программы.

Декодировщик используется для преобразования кода загружаемой программы в файл в формате ЯАС-программы или листинга. Преобразованный файл с исходным текстом программы может быть обработан компилятором.

3.2. Синтаксис файла с исходным текстом программы

Символические имена различаются по первым 16 символам. Имя операнда или метки не должно включать символы '~' (признак абсолютного операнда), ':' (признак метки), ';' (признак комментария), '¤', пробелы и знаки табуляции. Символы в именах вводятся с учетом регистра. Примеры корректных имен:

База_по_теплу
32-й_параметр
Delta

Инструкции могут вводиться как строчными, так и прописными символами: **jmp**, **Jmp**, **jMp** или **JMP**.
Если строка содержит метку, имя метки ограничивается символом ':', например:
Start: L ~M.3

Имя метки в инструкции перехода указывается без ':':

```
        jmp     Start
```

В общем случае строка инструкции может включать в себя следующие поля, разделенные одним или несколькими пробелами или знаками табуляции:

-поле метки (может отсутствовать), ограниченное справа символом ‘:’;

-поле инструкции (может отсутствовать);

-поле операнда (если инструкция имеет операнд);

-поле комментария, ограниченное слева символом ‘;’ (другие ограничители перед комментарием могут отсутствовать) (поле комментария также может отсутствовать). Внутри комментария (после символа ‘;’) могут быть любые символы.

Ниже приведен фрагмент программы, для которой список символических имен генерируется без ошибок.

```
Start:                                          ;Пример корректных синтаксических
;конструкций
Begin:          L       Mode;Пример комментария. Комментарий отделен от операнда
                ;только символом ;

                A               Errors

;;;
;;;
=       Result

;;;
end
```

4. Пример программы анализа событий

4.1. Схема узла учета и постановка задачи

Схема учета электроэнергии включает 18 счетчиков электроэнергии, установленных у абонентов, и 1 балансовый счетчик, измеряющий потребление электроэнергии всеми абонентами в сумме.

У абонентов установлены счетчики электроэнергии с импульсным выходом 4000 имп./КВтч. К импульсным выходам счетчиков подключены устройства типа МУР-1001.5СТ1, обеспечивающие подсчет числа импульсов с выхода счетчиков и передачу данных по сети 220 В по запросу от регистратора. Данные от счетчиков абонентов поступают в регистратор по силовой сети через модем МУР1001.9 RS232/220V.

В качестве балансового используется счетчик электроэнергии типа «Меркурий-200.02» с трансформатором тока 100/5. Связь между балансовым счетчиком и регистратором осуществляется по интерфейсу RS-485 через адаптер AD232TTL/485.

Требуется фиксировать факты превышения коммерческих потерь заданного порогового значения (10%). Коммерческие потери за контролируемый период оцениваются как относительная разность приращения показаний балансового счетчика и суммы приращений показаний счетчиков абонентов. Величина коммерческих потерь рассчитывается для 30-минутных интервалов.

4.2. Структура баз данных регистратора

Для решения поставленной задачи в регистраторе предусматривается ведение двух баз данных:

1. Ретроспективная база показания счетчиков абонентов и балансового счетчика. Опрос устройств предусматривается циклически, 2 раза в час. Структура ретроспективной базы приведена в табл. 4.1.

Табл. 4.1.

Структура ретроспективной базы данных

А-Логический номер устройства/Тип устройства/ Наименование параметра – комментарий	**Смещение относительно начала записи (HEX)**	**Тип и длина параметра**
Year – младшие 2 цифры года записи	0	BCD[1] 1
Month – месяц	1	BCD 1
Day – день	2	BCD 1
Hour – час	3	BCD 1
Min – минуты	4	BCD 1
StatusMUR – код состояния регистратора	5	Bin[2] 1
RxFlags0 –флаги связи с устройствами 0..7	6	Bin 1
RxFlags8 – флаги связи с устройствами 8..15	7	Bin 1
RxFlags16 – флаги связи с устройствами 16..23	8	Bin 1
A0/Счетчик_СТ1/Регистр – счетчик 1-го абонента	9	Bin 4
A1/Счетчик_СТ1/Регистр - счетчик 2-го абонента	0D	Bin 4
A2/Счетчик_СТ1/Регистр - счетчик 3-го абонента	11	Bin 4
A3/Счетчик_СТ1/Регистр - счетчик 4-го абонента	15	Bin 4
A4/Счетчик_СТ1/Регистр - счетчик 5-го абонента	19	Bin 4
A5/Счетчик_СТ1/Регистр - счетчик 6-го абонента	1D	Bin 4
A6/Счетчик_СТ1/Регистр - счетчик 7-го абонента	21	Bin 4
A7/Счетчик_СТ1/Регистр - счетчик 8-го абонента	25	Bin 4
A8/Счетчик_СТ1/Регистр - счетчик 9-го абонента	29	Bin 4
A9/Счетчик_СТ1/Регистр - счетчик 10-го абонента	2D	Bin 4
A10/Счетчик_СТ1/Регистр - счетчик 11-го абонента	31	Bin 4
A11/Счетчик_СТ1/Регистр - счетчик 12-го абонента	35	Bin 4
A12/Счетчик_СТ1/Регистр - счетчик 13-го абонента	39	Bin 4
A13/Счетчик_СТ1/Регистр - счетчик 14-го абонента	3D	Bin 4
A14/Счетчик_СТ1/Регистр - счетчик 15-го абонента	41	Bin 4
A15/Счетчик_СТ1/Регистр - счетчик 16-го абонента	45	Bin 4
A16/Счетчик_СТ1/Регистр - счетчик 17-го абонента	49	Bin 4
A17/Счетчик_СТ1/Регистр - счетчик 18-го абонента	4D	Bin 4
A18/Меркурий-200.02/Время – показания балансового счетчика: время	51	BCD 3
A18/Меркурий-200.02/Дата – показания балансового счетчика: дата	54	BCD 3
A18/Меркурий-200.02/Тариф_1 – показания балансового счетчика: энергия по тарифу 1	57	BCD 4
A18/Меркурий-200.02/Тариф_2 – показания балансового счетчика: энергия по тарифу 2	5B	BCD 4
A18/Меркурий-200.02/Суммарный – показания балансового счетчика: энергия по тарифам 1, 2, 3 и 4 в сумме	5F	BCD 4
CS – контрольная сумма записи	63	Bin 1

Примечания.

[1] BCD– Двоично-десятичный формат.

[2] Bin – Двоичное беззнаковое число.

2. Событийная база данных. Включает бит состояния и бит определенности единственного события, заключающего в результате сравнения величины коммерческих потерь с пороговым значением. Кроме того, в событийную базу необходимо включить величину коммерческих потерь. Так как периодичность ретроспективной базы равна 30 минутам, то и запуск процедуры анализа событий необходимо выполнять с такой же периодичностью.

4.3. Алгоритм работы программы анализа событий.

Расчет коммерческих потерь будет корректным при одновременном выполнении следующих условий:

- текущая и предыдущая записи ретроспективной базы данных корректны (т.е., контрольная сумма записей верна);

- биты флагов связи с устройствами в предыдущей и текущей записи сброшены, что соответствует наличию связи со всеми устройствами.

Дополнительно, корректность расчета обусловлена точностью показаний балансового счетчика. Счетчик электрической энергии «Меркурий-200.02» по интерфейсу RS-485 передает значения регистров накопленной энергии с точностью 0.01 КВтч или 10 Втч. С учетом коэффициента трансформатора тока (100/5) точность показаний балансного счетчика составит ±0.2 КВтч. Так как пороговое значение коммерческих потерь равно 10%, то (с учетом точности показаний балансного счетчика) считаем, что вычисления относительной разности приращения показаний балансного счетчика и суммы приращений показаний счетчиков абонентов будут корректны при приращениях показаний балансного счетчика более 4 КВтч. Или (с учетом продолжительности интервала времени для расчета потерь) при средней потребляемой мощности более 8 КВт.

Бит определенности события равен логическому «ИЛИ» трех перечисленных выше условий.

Единичное значение бита состояния события соответствует превышению порогового значения потерь, нулевое – если величина потерь меньше порогового значения.

Текст программы с комментариями приведен ниже.

```
Start:  clf   Result      ;Обнулить величину коммерческих потерь
        r     UnBalance   ;Сбросить признак события
        cb    Current     ;Проверка корректности текущей записи
        =     Define      ;
        cb    Prev        ;Проверка корректности предыдущей записи
        a     Define
        an    RxByte1c    ;Биты связи со всеми счетчиками в текущей и предыдущей записях
        an    RxByte2c    ;должны быть нулевыми
        an    RxByte3c
        an    RxByte1p
        an    RxByte2p
        an    RxByte3p
        =     Define
        lf    SummaC      ;Найти приращение показаний балансового счетчика
        -     SummaP
        *     KoeffB
        gt    MinValue
        a     Define
        =     Define      ;Если расчет потерь не корректен - выход
        jnr   Exit
        =f    Result      ;Записать приращение показаний балансового счетчика в Result
        lf    Count1p     ;Найти суммы предыдущих показаний счетчиков абонентов
        +     Count2p
        +     Count3p
        +     Count4p
        +     Count5p
        +     Count6p
        +     Count7p
        +     Count8p
        +     Count9p
        +     Count10p
        +     Count11p
        +     Count12p
        +     Count13p
        +     Count14p
        +     Count15p
        +     Count16p
        +     Count17p
```

```
                +       Count18p
                -       Count1c         ;Из суммы предыдущих показаний счетчиков абонетов
                -       Count2c         ;вычесть текущие показания
                -       Count3c
                -       Count4c
                -       Count5c
                -       Count6c
                -       Count7c
                -       Count8c
                -       Count9c
                -       Count10c
                -       Count11c
                -       Count12c
                -       Count13c
                -       Count14c
                -       Count15c
                -       Count16c
                -       Count17c
                -       Count18c        ;В ACC - отрицательное значение суммы приращений показаний
                *       KoeffCT         ;счетчиков абонентов
                +       Result          ;В ACC - абсолютная величина коммерческих потерь
                /       Result          ;В ACC – относительная величина коммерческих потерь
                =       Result
                gt      LimUnBalance    ;Сравнить относительную величину потерь с пороговым значением.
                =       UnBalance
Exit:           end
```

В программе использованы следующие символические имена:

Count1c..Count18c -показания счетчиков абонентов в текущей записи;
Count1p..Count18p -показания счетчиков абонентов в предыдущей записи;
Current -указатель на текущую запись;
Define -флаг определенности события;
LimUnBalance -предельное значение относительного дебаланса;
MinValue -пороговое значение приращения показаний балансового счетчика. Если значение приращения показаний балансового счетчика ниже порогового значения, расчет потерь не производится;
Prev -указатель на предыдущую запись;
Result -расчетная относительная величина потерь;
RxByte1c..RxByte3c -флаги связи со счетчиками в текущей записи;
RxByte1p..RxByte3p -флаги связи со счетчиками в предыдущей записи;
SummaC -показания балансового счетчика в текущей записи;
SummaP -показания балансового счетчика в предыдущей записи;
UnBalance -бит состояния события, устанавливается при превышении коммерческих потерь заданной величины.

После генерации списка символических имен, генерации структуры баз данных и определения символических имен будет создан de_-файл со следующей информацией:

```
Count10c    40310044 ;~FC.0.31.u4#B0/A10/Счетчик_CT1/Регистр
Count10p    48310044 ;~FP.0.31.u4#B0/A10/Счетчик_CT1/Регистр
Count11c    40350044 ;~FC.0.35.u4#B0/A11/Счетчик_CT1/Регистр
Count11p    48350044 ;~FP.0.35.u4#B0/A11/Счетчик_CT1/Регистр
Count12c    40390044 ;~FC.0.39.u4#B0/A12/Счетчик_CT1/Регистр
Count12p    48390044 ;~FP.0.39.u4#B0/A12/Счетчик_CT1/Регистр
Count13c    403D0044 ;~FC.0.3D.u4#B0/A13/Счетчик_CT1/Регистр
Count13p    483D0044 ;~FP.0.3D.u4#B0/A13/Счетчик_CT1/Регистр
Count14c    40410044 ;~FC.0.41.u4#B0/A14/Счетчик_CT1/Регистр
Count14p    48410044 ;~FP.0.41.u4#B0/A14/Счетчик_CT1/Регистр
Count15c    40450044 ;~FC.0.45.u4#B0/A15/Счетчик_CT1/Регистр
Count15p    48450044 ;~FP.0.45.u4#B0/A15/Счетчик_CT1/Регистр
Count16c    40490044 ;~FC.0.49.u4#B0/A16/Счетчик_CT1/Регистр
Count16p    48490044 ;~FP.0.49.u4#B0/A16/Счетчик_CT1/Регистр
Count17c    404D0044 ;~FC.0.4D.u4#B0/A17/Счетчик_CT1/Регистр
Count17p    484D0044 ;~FP.0.4D.u4#B0/A17/Счетчик_CT1/Регистр
Count18c    40510003 ;~FC.0.51.b3#B0/A18/Меркурий-200.02/Время
```

```
Count18p        48510003 ;~FP.0.51.b3#B0/A18/Меркурий-200.02/Время
Count1c         400D0044 ;~FC.0.D.u4#B0/A1/Счетчик_CT1/Регистр
Count1p         480D0044 ;~FP.0.D.u4#B0/A1/Счетчик_CT1/Регистр
Count2c         40110044 ;~FC.0.11.u4#B0/A2/Счетчик_CT1/Регистр
Count2p         48110044 ;~FP.0.11.u4#B0/A2/Счетчик_CT1/Регистр
Count3c         40150044 ;~FC.0.15.u4#B0/A3/Счетчик_CT1/Регистр
Count3p         48150044 ;~FP.0.15.u4#B0/A3/Счетчик_CT1/Регистр
Count4c         40190044 ;~FC.0.19.u4#B0/A4/Счетчик_CT1/Регистр
Count4p         48190044 ;~FP.0.19.u4#B0/A4/Счетчик_CT1/Регистр
Count5c         401D0044 ;~FC.0.1D.u4#B0/A5/Счетчик_CT1/Регистр
Count5p         481D0044 ;~FP.0.1D.u4#B0/A5/Счетчик_CT1/Регистр
Count6c         40210044 ;~FC.0.21.u4#B0/A6/Счетчик_CT1/Регистр
Count6p         48210044 ;~FP.0.21.u4#B0/A6/Счетчик_CT1/Регистр
Count7c         40250044 ;~FC.0.25.u4#B0/A7/Счетчик_CT1/Регистр
Count7p         48250044 ;~FP.0.25.u4#B0/A7/Счетчик_CT1/Регистр
Count8c         40290044 ;~FC.0.29.u4#B0/A8/Счетчик_CT1/Регистр
Count8p         48290044 ;~FP.0.29.u4#B0/A8/Счетчик_CT1/Регистр
Count9c         402D0044 ;~FC.0.2D.u4#B0/A9/Счетчик_CT1/Регистр
Count9p         482D0044 ;~FP.0.2D.u4#B0/A9/Счетчик_CT1/Регистр
Current         D0000000 ;~PC.0
Define          10000000 ;~DC.0
LimUnBalance    C2000001 ;~C.0.1
KoeffB          C2000002 ;~C.0.2
KoeffCT         C2005032 ;~C.00025
MinValue        C0000014 ;~C.4.0
Prev            D8000000 ;~PP.0
Result          28000000 ;~R.0
RxByte1c        500600FF ;~BC.0.6&FF#B0/RxFlags0+0&FF
RxByte1p        580600FF ;~BP.0.6&FF#B0/RxFlags0+0&FF
RxByte2c        500700FF ;~BC.0.7&FF#B0/RxFlags8+0&FF
RxByte2p        580700FF ;~BP.0.7&FF#B0/RxFlags8+0&FF
RxByte3c        50080003 ;~BC.0.8&3#B0/RxFlags16+0&3
RxByte3p        58080003 ;~BP.0.8&3#B0/RxFlags16+0&3
SummaC          405F0004 ;~FC.0.5F.b4#B0/A18/Меркурий-200.02/Суммарный
SummaP          485F0004 ;~FP.0.5F.b4#B0/A18/Меркурий-200.02/Суммарный
UnBalance       00000000 ;~EC.0
¤Exit           3D000000
¤Start          00000000
```

После компиляции программа загружается в регистратор.