

```

0001 FUNCTION int_to_sec : TIME
0002 VAR_INPUT
0003 in: INT;
0004 END_VAR
0005 VAR
0006 END_VAR
0001 int_to_sec := DINT_TO_TIME(INT_TO_DINT(in) * 1000);
sec_to_int (FUN-ST)
0001 FUNCTION sec_to_int : INT
0002 VAR_INPUT
0003 in: TIME;
0004 END_VAR
0005 VAR
0006 END_VAR
0001 sec_to_int := DINT_TO_INT(TIME_TO_DINT(in) / 1000);
AirValve (FB-ST)
0001 FUNCTION_BLOCK AirValve
0002 VAR_INPUT
0003 id: INT ; (*номер устройства*)
0004 rst_err: BOOL ;
0005 fire: BOOL; (*в этом режиме игнорируются все аварии*)
0006 cmd_open: BOOL ; (*открыть*)
0007 pos_open: BOOL ; (*положение открыто*)
0008 tm_op_vlv: INT ; (*Время на открытие задвижек в секундах*)
0009 ignr_md : BOOL ; (*Режим игнорирования концевика*)
0010 END_VAR
0011 VAR_OUTPUT
0012 alrm_open: BOOL ; (*авария открытия задвижки*)
0013 err: BOOL ;
0014 dl_op: INT ;
0015 END_VAR
0016
0017 VAR_IN_OUT
0018 err_arr: ARRAY [0..ALRM_NMBR] OF BOOL;
0019 END_VAR
0020
0021 VAR
0022 delay_op: TON;
0023 END_VAR
0001 delay_op(
0002 IN := cmd_open AND NOT pos_open,
0003 PT := int_to_sec(tm_op_vlv)
0004 );
0005 IF delay_op.Q AND NOT fire AND NOT ignr_md THEN
0006 alrm_open := TRUE;
0007 END_IF
0008
0009 dl_op := sec_to_int(delay_op.ET);
0010
0011 IF rst_err THEN
0012 alrm_open := FALSE;
0013 END_IF
0014
0015 err := alrm_open;
0016 err_arr[id] := err;
AnalogValve (FB-ST)
0001 FUNCTION_BLOCK AnalogValve
0002 VAR_INPUT
0003 id: INT ; (*номер устройства*)
0004 pid_value: REAL ; (*управляющее значение от ПИД регулятора*)
0005 ai_pos: REAL ; (*положение клапана от модуля AI*)
0006 rst_err: BOOL ; (*сброс аварий*)

```

```

0007 END_VAR
0008 VAR_OUTPUT
0009 pos: REAL ; (*положение клапана в %*)
0009 pos: REAL ; (*положение клапана в %*)
0010 val: REAL ; (*управляющее значение в %*)
0011 ao_val: INT ; (*управляющее значение положения клапана на модуль AO*)
0012 err: BOOL ;
0013 delay_vlv : INT;
0014 mismatch : BOOL;
0015 END_VAR
0016 VAR_IN_OUT
0017 err_arr: ARRAY [0..ALRM_NMBR] OF BOOL;
0018 END_VAR
0019 VAR
0020 max_snsr_val: REAL :=100.0; (*максимальное значение, которое принимает датчик*)
0021 adc_res: INT :=100; (*разрешение ацп модуля 32767*)
0022 delay_time: TIME :=T#2m;
0023 delay: TON ;
0024 mismatch_value: REAL :=10; (*величина рассогласования*)
0025 END_VAR
0001 (*
0002 pos := INT_TO_REAL(ai_pos) * max_snsr_val / adc_res;
0003
0004 (*округление до целых*)
0005 IF pos > INT_TO_REAL(REAL_TO_INT(pos)) + 0.5 THEN
0006 pos := INT_TO_REAL(REAL_TO_INT(pos) + 1);
0007 ELSE
0008 pos := INT_TO_REAL(REAL_TO_INT(pos));
0009 END_IF
0010 val := ABS(pid_value);
0011 ao_val := REAL_TO_INT(pid_value * adc_res / max_snsr_val);
0012 *)
0013
0014 pos := ai_pos;
0015
0016
0017 val := ABS(pid_value);
0018 ao_val := REAL_TO_INT(pid_value * adc_res / max_snsr_val);
0019
0020
0021 delay(
0022 IN := ABS(pid_value - pos) > mismatch_value,
0023 PT := delay_time,
0024 );
0025
0026 IF delay.Q THEN
0027 mismatch := TRUE;
0028 END_IF
0029
0030 delay_vlv := sec_to_int(delay.ET);
0031 IF delay.Q AND NOT fire THEN
0032 err := TRUE;
0033 END_IF
0034
0035 IF rst_err THEN
0036 err := FALSE;
0037 mismatch := FALSE;
0038 END_IF
0039
0040 err_arr[ID] := err;

```

```

check_break (FB-ST)
0001 FUNCTION_BLOCK check_break
0002 VAR_INPUT
0003 sens_state : WORD;
0004 END_VAR
0005 VAR_OUTPUT
0006 err : BOOL ;
0007 END_VAR
0008 VAR
0009 END_VAR
0001 IF sens_state <> 0 THEN
0002 err := TRUE;
0003 ELSE
0004 err := FALSE;
0005 END_IF
freq_lim (FB-ST)
0001 FUNCTION_BLOCK freq_lim
0002 VAR_INPUT
0003 LOW_FREQ_LIM : INT;
0004 HIGH_FREQ_LIM : INT;
0005 freq_in : INT;
0006 END_VAR
0007 VAR_OUTPUT
0008 freq_out : WORD;
0009 END_VAR
0010 VAR
0011 LOW_MD_LIM : INT := 0;
0012 HIGH_MD_LIM : INT := 1000;
0013 END_VAR
0001 (*0 - 20Гц
0002 1000 - 50Гц
0003
0004 =>40Гц - (1000-0)/(50-20)*(40-20)*)
0005 freq_out := (HIGH_MD_LIM - LOW_MD_LIM)/(HIGH_FREQ_LIM - LOW_FREQ_LIM)*(freq_in -
LOW_FREQ_LIM);
hmi_in_out (FB-ST)
0001 FUNCTION_BLOCK hmi_in_out
0002 VAR_INPUT
0003 arr_in_out_coil : ARRAY [1..MAX_NMBR_ITEMS] OF BOOL;
0004 arr_sf_in_out_coil : ARRAY [1..MAX_NMBR_ITEMS] OF BOOL;
0005 arr_in_out_reg : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0006 arr_sf_in_out_reg : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0007 check_by_var : WORD;
0008 END_VAR
0009 VAR_OUTPUT
0010 arr_r_state : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0011 arr_w_state : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0012 END_VAR
0013 VAR
0014 tmr_rstr : TON;
0015 trg_rstr : R_TRIG;
0016 i : INT;
0017 END_VAR
0001 IF (check_by_var = 0) THEN (*Разрешена только запись в панель*)
0002 IF NOT rst_err THEN
0003 tmr_rstr(IN:=TRUE, PT:=t#5s);
0004 ELSE
0005 tmr_rstr(IN:=FALSE);
0006 END_IF
0007 IF tmr_rstr.Q THEN
0008 h_iw2_state := 255;
0009 ELSE

```

```

0010 FOR i := 1 TO MAX_NMBR_ITEMS DO
0011 arr_w_state[i] := 255;
0012 END_FOR;
0013 END_IF
0014 ELSE
0015 FOR i := 1 TO MAX_NMBR_ITEMS DO
0016 arr_r_state[i] := 255;
0017 END_FOR;
0018
0019 FOR i := 1 TO MAX_NMBR_ITEMS DO
0020 arr_sf_in_out_coil[i] := arr_in_out_coil[i];
0021 END_FOR;
0022
0023 FOR i := 1 TO MAX_NMBR_ITEMS DO
0024 arr_sf_in_out_reg[i] := arr_in_out_reg[i];
0025 END_FOR;
0026 END_IF
0001 FUNCTION_BLOCK Motor
0002 VAR_INPUT
0003 id : INT ; (*номер устройства*)
0004 volt_check : BOOL ;
0005 brkr : BOOL ;
0006 auto : BOOL ;
0007 sft_brkr : BOOL := TRUE;
0008 run : BOOL ;
0009 t_ctrl : BOOL := TRUE;
0010 diff_ctrl : BOOL ;
0011 msg_lnch : BOOL ;
0012 rst_err : BOOL ;
0013 fire : BOOL ;
0014 freq_err : BOOL ;
0015 tm_diff_err : INT ;
0016 state_vlv : BOOL ;
0017 END_VAR
0018 VAR_OUTPUT
0019 cmd_open : BOOL;
0020 END_VAR
0021 VAR_IN_OUT
0022 err_arr: ARRAY [0..ALRM_NMBR] OF BOOL;
0023 END_VAR
0024 VAR
0025 tmr_diff_err : TON ;
0026 tmr_strt_mtr : TON ;
0027 diff_err: BOOL ;
0028 strt_err: BOOL ;
0029 END_VAR
0001 err_act;
0002
0003 IF msg_lnch AND NOT err_arr[id] THEN
0004 cmd_open := TRUE;
0005 ELSE
0006 cmd_open := FALSE;
0007 END_IF
err_act (ST)
0001 tmr_diff_err(
0002 in := run AND NOT diff_ctrl,
0003 pt := int_to_sec(tm_diff_err)
0004 );
0005
0006 tmr_strt_mtr(
0007 in := cmd_open AND NOT run,
0008 pt := int_to_sec(tm_strt_mtr)

```

```

0009 );
0010
0011 IF tmr_diff_err.Q AND NOT fire THEN
0012 diff_err := TRUE;
0013 END_IF
0014
0015 IF tmr_strt_mtr.Q AND NOT fire THEN
0016 strt_err := TRUE;
0017 END_IF
0018 IF rst_err THEN
0019 diff_err := FALSE;
0020 strt_err := FALSE;
0021 END_IF
0022
0023 IF NOT fire THEN
0024 err_arr[id+1] := diff_err;
0025 err_arr[id+2] := NOT brkr;
0026 err_arr[id+3] := NOT sft_brkr;
0027 err_arr[id+4] := NOT t_ctrl;
0028 err_arr[id+5] := freq_err;
0029 err_arr[id+6] := NOT volt_check;
0030 err_arr[id+7] := strt_err;
0031 err_arr[id+8] := FALSE;
0032 err_arr[id+9] := FALSE;
0033 err_arr[id+10] := FALSE;
0034 END_IF
0035
0036 err_arr[id] := err_arr[id+1] OR err_arr[id+2] OR err_arr[id+3] OR
0037 err_arr[id+4] OR err_arr[id+5] OR err_arr[id+6] OR
0038 err_arr[id+7] ;
Operating (FB-ST)
0001 FUNCTION_BLOCK Operating
0002 VAR_INPUT
0003 t_q : BOOL ; (*Выход плк, запускающий устройство*)
0004 END_VAR
0005 VAR_IN_OUT
0006 t_h1_work : DWORD ; (*Наработка в часах*)
0007 END_VAR
0008 VAR
0009 tm : TON ; (*Общий циклический таймер для всех устройств*)
0010 END_VAR
0001 tm(PT := T#1h, IN := NOT tm.Q);
0002
0003 t_h1_work := t_h1_work + BOOL_TO_DWORD(tm.Q AND t_q);
Rotation (FB-ST)
0001 FUNCTION_BLOCK Rotation
0002 VAR_INPUT
0003 alrm_mtr_1 : BOOL;
0004 alrm_mtr_2 : BOOL;
0005 man_strt_mtr_1 : BOOL;
0006 man_strt_mtr_2 : BOOL;
0007 mtr_enbl_1 : BOOL;
0008 mtr_enbl_2 : BOOL;
0009 sp_tm_rttm : TIME;
0010 tm_dl_rttm : TIME;
0011 END_VAR
0012 VAR_OUTPUT
0013 strt_mtr_1 : BOOL;
0014 strt_mtr_2 : BOOL;
0015 END_VAR
0016 VAR
0017 Blinker : BLINK;

```

```

0018 tmr_strt: TON;
0019 trg_strt_mtr_1: R_TRIG;
0020 trg_strt_mtr_2: R_TRIG;
0021 trg_stop_mtr_1: R_TRIG;
0022 trg_stop_mtr_2: R_TRIG;
0023 mtr_1 : BOOL;
0024 mtr_2 : BOOL;
0025 test_et : INT;
0026 END_VAR
0001 IF alrm_mtr_1 AND alrm_mtr_2 THEN
0002 mtr_2 := FALSE;
0003 mtr_1 := FALSE;
0004 ELSIF alrm_mtr_1 AND NOT alrm_mtr_2 THEN
0005 mtr_2 := TRUE;
0006 mtr_1 := FALSE;
0007 ELSIF alrm_mtr_2 AND NOT alrm_mtr_1 THEN
0008 mtr_1 := TRUE;
0009 mtr_2 := FALSE;
0010 ELSE
0011 IF man_strt_mtr_1 THEN
0012 mtr_1 := TRUE;
0013 mtr_2 := FALSE;
0014 ELSIF man_strt_mtr_2 THEN
0015 mtr_2 := TRUE;
0016 mtr_1 := FALSE;
0017 ELSE
0018 IF mtr_enbl_1 AND mtr_enbl_2 THEN
0019 Blinker(
0020 ENABLE:= TRUE,
0021 TIMELOW:= sp_tm_rttm,
0022 TIMEHIGH:= sp_tm_rttm,
0023 );
0024 IF blinker.OUT THEN
0025 mtr_1 := TRUE;
0026 mtr_2 := FALSE;
0027 ELSE
0028 mtr_1 := FALSE;
0029 mtr_2 := TRUE;
0030 END_IF
0031
0032 ELSIF mtr_enbl_1 AND NOT mtr_enbl_2 THEN
0033 mtr_1 := TRUE;
0034 mtr_2 := FALSE;
0035
0036 ELSIF mtr_enbl_2 AND NOT mtr_enbl_1 THEN
0037 mtr_1 := FALSE;
0038 mtr_2 := TRUE;
0039 ELSE
0040 mtr_1 := FALSE;
0041 mtr_2 := FALSE;
0042 END_IF
0043 END_IF
0044 END_IF
0045
0046 trg_strt_mtr_1(CLK:=mtr_1);
0047 trg_strt_mtr_2(CLK:=mtr_2);
0048
0049 trg_stop_mtr_1(CLK:=mtr_1);
0050 trg_stop_mtr_2(CLK:=mtr_2);
0051 (*
0052 IF trg_stop_mtr_1.Q OR trg_stop_mtr_1.Q THEN
0053 tmr_strt(IN:=FALSE);

```

```

0054 END_IF*)
0055
0056 IF NOT trg_strt_mtr_1.Q OR NOT trg_strt_mtr_2.Q OR NOT trg_stop_mtr_1.Q OR NOT
trg_stop_mtr_2.Q THEN
0057 tmr_strt(IN:=TRUE, PT:=tm_dl_rtt);
0058 END_IF
0059
0060 test_et:=sec_to_int(tmr_strt.ET);
0061
0062 IF trg_strt_mtr_1.Q OR trg_strt_mtr_2.Q OR trg_stop_mtr_1.Q OR trg_stop_mtr_2.Q
THEN
0063 tmr_strt(IN:=FALSE);
0064 END_IF
0065
0066 IF tmr_strt.Q THEN
0067 strt_mtr_1 := mtr_1;
0068 strt_mtr_2 := mtr_2;
0069 ELSE
0070 strt_mtr_1 := FALSE;
0071 strt_mtr_2 := FALSE;
0072 END_IF
0073
0074
0075
SensorT (FB-ST)
0001 FUNCTION_BLOCK SensorT
0002 VAR_INPUT
0003 id: INT ; (*номер устройства*)
0004 input: REAL ;
0005 use_defval: BOOL ; (*разрешение на использование значения по-умолчанию*)
0006
0007 rst_err: BOOL ; (*сброс аварий*)
0008 fire: BOOL ; (*в этом режиме игнорируются все аварии*)
0009 max_val: REAL :=100; (*максимальное значение, которое может принимать датчик*)

0009 max_val: REAL :=100; (*максимальное значение, которое может принимать датчик*)
0010 min_val: REAL :=-40; (*минимальное значение, которое может принимать датчик*)
0011 END_VAR
0012
0013 VAR_OUTPUT
0014 value: REAL ;
0015 err: BOOL ;
0016 END_VAR
0017
0018 VAR_IN_OUT
0019 err_arr: ARRAY [0..ALRM_NMBR] OF BOOL ;
0020 END_VAR
0021
0022 VAR
0023 def_val: REAL ;
0024 END_VAR
0001 value := input;
0002
0003
0004 IF (value < min_val OR value > max_val) AND NOT fire THEN
0005 err := TRUE;
0006 ELSE
0007 def_val := value;
0008 END_IF
0009
0010 IF rst_err THEN
0011 err := FALSE;

```

```

0012 END_IF
0013
0014 IF use_defval AND err THEN
0015 value := def_val;
0016 END_IF
0017
0018 err_arr[id] := err;

Htr (PRG-ST)
0001 PROGRAM Htr
0002 VAR_INPUT
0003 gist: REAL := 3; (*гистерезис отклонения от уставки в обратной*)
0004 rst_err: BOOL ;
0005 END_VAR
0006
0007 VAR
0008 rttm : Rotation;
0009 reg: PID ; (*модуль ПИД регулятора*)
0010 valve: AnalogValve ; (*модуль 3х ходового клапана*)
0011 pmp: Motor ; (*насос*)
0012 pmp_r: Motor ; (*насос*)
0013 oprtng_pmp: Operating; (*Наработка основной*)
0014 oprtng_pmp_r: Operating; (*Наработка резервный*)
0015 cntr_preheat : CTU;
0016 tmr_preheat : TON;
0017 out_temp_pnts: ARRAY [0..4] OF REAL;
0018 invrs_temp_pnts: ARRAY [0..4] OF REAL;
0019 i: INT ;
0020 tmr_not_auto: TON ;
0021 END_VAR
0022
0023 VAR_OUTPUT
0024 invrs_temp: SensorT ; (*датчик температуры обратной воды*)
0025 wntr_ready: BOOL ; (*сигнал готовности к запуску приточки зимой*)
0026 END_VAR
0001 init_sensors;
0002 alrm_gen;
0003 oprtng_time;
0004 rotation;
0005
0006 IF winter_mode THEN
0007 winter_act;
0008 init_actors;
0009 calc_invrt_sp;
0010 ELSE
0011 summer_act;
0012 END_IF
alrm_gen (ST)
0001 (*Авария в случае отсутствия режима "Авто" или не работы в режиме "Ручной"*)
0002 tmr_not_auto.in := NOT (di_auto_pump_p OR di_run_pump_p OR
0003 di_auto_pump_p_r OR di_run_pump_p_r)
0004 AND winter_mode;
0005
0006 IF htr_invrs_temp_err_sp > htr_invrs_temp THEN
0007 alrm_invrs_temp := TRUE;
0008 END_IF
0009
0010 tmr_not_auto(
0011 pt := int_to_sec(tmr_not_auto),
0012 q => warn_not_auto
0013 );
0014

```



```

0015 cur_tm_not_auto := sec_to_int(tmr_not_auto.ET);
0016
0017 (*Авария по превышению прогревов в сутки*)
0018 cntr_preheat(
0019 CU:= reg_state = PREHEAT,
0020 RESET:= tmr_preheat.Q OR rst_err,
0021 PV:= max_cnt_preheat,
0022 CV=> cnt_preheat
0023 );
0024
0025 IF cnt_preheat >= 1 THEN
0026 tmr_preheat(
0027 IN:= TRUE,
0028 PT:= tm_cntr_preheat
0029 );
0030 ELSE
0031 tmr_preheat(IN:= FALSE);
0032 END_IF
0033
0034 IF cntr_preheat.Q THEN
0035 alm_cnt_preheat := TRUE;
0036 END_IF
0037
0038 IF do_strt_pump_p AND do_strt_pump_pr THEN
0039 alm_cnt_pmp := TRUE;
0040 END_IF
calc_invrt_sp (ST)
0001 out_temp_pnts[ 0 ] := htr_out_temp_p1 ;
0002 out_temp_pnts[ 1 ] := htr_out_temp_p2 ;
0003 out_temp_pnts[ 2 ] := htr_out_temp_p3 ;
0004 out_temp_pnts[ 3 ] := htr_out_temp_p4 ;
0005 out_temp_pnts[ 4 ] := htr_out_temp_p5 ;
0006
0007 invrs_temp_pnts[ 0 ] := htr_invrs_temp_p1 ;
0008 invrs_temp_pnts[ 1 ] := htr_invrs_temp_p2 ;
0009 invrs_temp_pnts[ 2 ] := htr_invrs_temp_p3 ;
0010 invrs_temp_pnts[ 3 ] := htr_invrs_temp_p4 ;
0011 invrs_temp_pnts[ 4 ] := htr_invrs_temp_p5 ;
0012
0013
0014 (*расчёт уставки температуры обратной воды по графику из 5 точек*)
0015 IF sup_in_temp >= out_temp_pnts[0] THEN
0016 htr_invrs_temp_sp := invrs_temp_pnts[0];
0017 ELSIF sup_in_temp <= out_temp_pnts[4] THEN
0018 htr_invrs_temp_sp := invrs_temp_pnts[4];
0019 ELSE
0020 FOR i := 1 TO 4 DO
0021 IF sup_in_temp = out_temp_pnts[i] THEN
0022 htr_invrs_temp_sp := invrs_temp_pnts[i];
0023 END_IF
0024 IF sup_in_temp > out_temp_pnts[i] AND
0025 sup_in_temp < out_temp_pnts[i - 1] THEN
0026 htr_invrs_temp_sp := (invrs_temp_pnts[i] - invrs_temp_pnts[i - 1]) *
0027 ABS(sup_in_temp - out_temp_pnts[i - 1]) /
0028 ABS(out_temp_pnts[i] - out_temp_pnts[i - 1]) +
0029 invrs_temp_pnts[i-1];
0030 END_IF
0031 END_FOR
0032 END_IF;
init_actors (ST)
0001 valve(
0002 id := ID_HTR_VLV,

```

```

0003 rst_err := rst_err,
0004 pid_value := reg.Y,
0005 ai_pos := ai_state_y1,
0006 err_arr := alrm_arr,
0007 pos => htr_vlv_pos,
0008 val => htr_vlv_value,
0009 delay_vlv => cur_tm_delay_vlv,
0010 mismatch => warn_mismatch
0011
0012 );
0013
0014 ao_y1 := valve.ao_val * 10;
0015
0016 pmp(
0017 id := ID_HTR_PMP,
0018 rst_err := rst_err,
0019 volt_check := di_phase_ctrl,
0020 brkr := NOT di_alrm_pump_p,
0021 auto := di_auto_pump_p,
0022 run := di_run_pump_p,
0023 diff_ctrl := TRUE,
0024 tm_diff_err := tm_diff_err,
0025 fire := fire,
0026 err_arr := alrm_arr,
0027 state_vlv := TRUE
0028 );
0029
0030 pmp_r(
0031 id := ID_HTR_PMP_R,
0032 rst_err := rst_err,
0033 volt_check := di_phase_ctrl,
0034 brkr := NOT di_alrm_pump_p_r,
0035 auto := di_auto_pump_p_r,
0036 run := di_run_pump_p_r,
0037 diff_ctrl := TRUE,
0038 tm_diff_err := tm_diff_err,
0039 fire := fire,
0040 err_arr := alrm_arr,
0041 state_vlv := TRUE
0042 );
0043
0044 reg(
0045 KP:= htr_vlv_kp,
0046 TN:= htr_vlv_tn,
0047 TV:= htr_vlv_tv,
0048 Y_MIN:= PID_Y_MIN,
0049 Y_MAX:= PID_Y_MAX,
0050 );
init_sensors (ST)
0001 invrs_temp(
0002 id := ID_HTR_INVRS_TEMP,
0003 rst_err := rst_err,
0004 input := ai_invr_temp,
0005 fire := fire,
0006 def_val := ,
0007 err_arr := alrm_arr,
0008 value => htr_invrs_temp,
0009 err =>
0010 );
oprtnng_time (ST)
0001 oprtnng_pmp(
0002 t_q := di_run_pump_p,

```

```

0003 t_h1_work := tm_oprtn_pmp
0004 );
0005
0006 oprtng_pmp_r(
0007 t_q := di_run_pump_p_r,
0008 t_h1_work := tm_oprtn_pmp_r
0009 );
rotation (ST)
0001 IF (pmp.auto OR pmp_r.auto) AND winter_mode THEN
0002 rttt(
0003 alrm_mtr_1:= alrm_htr_pmp OR di_alrm_pump_p,
0004 alrm_mtr_2:= alrm_htr_pmp_r OR di_alrm_pump_p_r,
0005 man_strt_mtr_1:= FALSE,
0006 man_strt_mtr_2:= FALSE,
0007 mtr_enbl_1:= pmp.auto,
0008 mtr_enbl_2:= pmp_r.auto,
0009 sp_tm_rttt:= tm_rttt_pmp,
0010 tm_dl_rttt:= tm_dl_rttt_pmp,
0011 strt_mtr_1=> do_strt_pump_p,
0012 strt_mtr_2=> do_strt_pump_pr);
0013 ELSE
0014 do_strt_pump_p := FALSE;
0015 do_strt_pump_pr := FALSE;
0016 END_IF
summer_act (ST)
0001 pmp.msg_lrch := FALSE;
0002 pmp_r.msg_lrch := FALSE;
0003
0004 reg_state := NO_REG;
winter_act (ST)
0001 IF NOT alrm_htr_pmp AND NOT alrm_cnt_pmp THEN
0002 pmp.msg_lrch := TRUE;
0003 ELSE
0004 pmp.msg_lrch := FALSE;
0005 END_IF
0006
0007 IF NOT alrm_htr_pmp_r AND NOT alrm_cnt_pmp THEN
0008 pmp_r.msg_lrch := TRUE;
0009 ELSE
0010 pmp_r.msg_lrch := FALSE;
0011 END_IF
0012
0013 wntr_ready := htr_invrs_temp > htr_invrs_temp_sp - gist;
0014
0015 CASE reg_state OF
0016 NO_REG:
0017 reg.MANUAL := TRUE;
0018 reg.Y_MANUAL := 0;
0019 MAINTAIN:
0020 reg.ACTUAL := htr_invrs_temp;
0021 reg.SET_POINT := htr_invrs_temp_sp;
0022 reg.MANUAL := FALSE;
0023 PREHEAT:
0024 reg.MANUAL := TRUE;
0025 reg.Y_MANUAL := 100;
0026 WAIT_REG:
0027 reg.MANUAL := TRUE;
0028 reg.Y_MANUAL := 100;
0029 REGULATOR:
0030 reg.ACTUAL := sup_out_temp;
0031 IF hmi_sup_out_temp_sp < 16 THEN
0032 reg.SET_POINT := 16;

```

```

0033 ELSE
0034 reg.SET_POINT := hmi_sup_out_temp_sp;
0035 END_IF
0036 reg.MANUAL := FALSE;
0037
0038 END_CASE
Sup (PRG-ST)
0001 PROGRAM Sup
0002 VAR_INPUT
0003 rst_err: BOOL ;
0004 END_VAR
0005 VAR
0006 y: AirValve; (*Воздушная заслонка*)
0007 fan: Motor; (*Двигатель вентилятора*)
0008 fan_r: Motor; (*Двигатель вентилятора*)
0009 oprtng_p: Operating; (*Наработка основной*)
0010 oprtng_p_r: Operating; (*Наработка резервный*)
0011 temp_in: SensorT ; (*датчик температуры на входе приточки*)
0012 temp_out: SensorT ; (*датчик температуры на выходе приточки*)
0013
0014 tmr_strt_p: TON ;
0015 tmr_sup_strt: TON ;
0016 freq_lim_p: freq_lim ;
0017 freq_lim_p_r: freq_lim ;
0018 tmr_freq_reg_lvl_1 : TOF;
0019 man_freq_reg_lvl_1 : BOOL;
0020 tmr_freq_reg_lvl_2 : TOF;
0021 man_freq_reg_lvl_2 : BOOL;
0022 rttm : Rotation;
0023 tmr_ignr_vlv_mode : TON;
0024 tmr_alrm_sup_temp : TON;
0025 trg_strt_sup: R_TRIG ;
0026 trg_stop_sup: F_TRIG ;
0027 END_VAR
0001 init_sensors;
0002 init_actors;
0003 alrm_gen;
0004 freq_reg;
0005 rotation;
0006 oprtng_time;
0007
0008
0009 trg_strt_sup(CLK:=strt_sup);
0010 trg_stop_sup(CLK:=strt_sup);
0011 tmr_strt_p(pt := int_to_sec(tm_strt_p));
0012 tmr_sup_strt(pt := int_to_sec(tm_sup_strt));
0013
0014 CASE sys_state OF
0015 OFF:
0016 case_off;
0017 reg_state := MAINTAIN;
0018 START_HEATING:
0019 case_strt_heat;
0020 reg_state := PREHEAT;
0021 START_OP_VLV:
0022 case_op_vlv;
0023 reg_state := MAINTAIN;
0024 START_LNCH_FAN:
0025 case_lnch_fan;
0026 reg_state := MAINTAIN;
0027 RUN_REG:
0028 case_run_reg;

```

```

0029 reg_state := REGULATOR;
0030 STOPPING;
0031 case_stopping;
0032 reg_state := MAINTAIN;
0033 END_CASE
almr_gen (ST)
0001 IF prtct_air THEN
0002 alrm_ts := TRUE;
0003 END_IF
0004
0005 tmr_almr_sup_temp(
0006 IN := sup_out_temp < sup_out_temp_err,
0007 PT := t#10s
0008 );
0009
0010 IF winter_mode AND tmr_almr_sup_temp.Q THEN
0011 alrm_sup_temp := TRUE;
0012 END_IF
0013
0014 IF do_strt_fan_p AND do_strt_fan_p_r THEN
0015 alrm_cnt_freq := TRUE;
0016 END_IF
0017
0018 prmsn_sup := NOT (alrm_invrs_temp OR
0019 alrm_sup_temp OR
0020 alrm_preheat OR
0021 alrm_ts OR
0022 fire OR
0023 alrm_cnt_freq OR
0024 alrm_sup_vlv OR
0025 alrm_brk_sens_invrt OR
0026 alrm_brk_sens_in OR
0027 (alrm_sup_fan_diff AND alrm_sup_fan_r_diff) OR
0028 (alrm_sup_fan_brkr AND alrm_sup_fan_r_brkr) OR
0029 (alrm_sup_fan_strt AND alrm_sup_fan_r_strt) OR
0030 (alrm_htr_pmp AND alrm_htr_pmp_r)) AND
0031 (di_auto_fan_p OR di_auto_fan_p_r);
case_lnch_fan (ST)
0001 IF di_pds_p THEN
0002 sys_state := RUN_REG;
0003 END_IF
0004
0005 IF NOT prmsn_sup OR trg_stop_sup.Q THEN
0006 sys_state := STOPPING;
0007 END_IF
case_off (ST)
0001 tmr_strt_p.in := FALSE;
0002 tmr_sup_strt.in := FALSE;
0003
0004 fan.msg_lnch := FALSE;
0005 fan_r.msg_lnch := FALSE;
0006
0007
0008 IF trg_strt_sup.Q THEN
0009 IF winter_mode THEN
0010 sys_state := START_HEATING;
0011 ELSE
0012 sys_state := START_OP_VLV;
0013 END_IF
0014 END_IF
case_op_vlv (ST)
0001 (*Открываем заслонки, частота 0*)

```

```

0002 IF (NOT hmi_set_strt_mode AND mb_di_pds_fan_v01_1) OR hmi_set_strt_mode THEN
0003 fan.msg_lnch := TRUE;
0004 fan_r.msg_lnch := TRUE;
0005 ELSE
0006 fan.msg_lnch := FALSE;
0007 fan_r.msg_lnch := FALSE;
0008 END_IF
0009
0010 (*Открываем заслонки при режиме игнорирования концевиков, частота 0*)
0011 IF fan.msg_lnch OR fan_r.msg_lnch THEN
0012 IF hmi_ignr_vlv_mode THEN
0013 tmr_ignr_vlv_mode(
0014 IN:= do_strt_fan_p OR do_strt_fan_p_r,
0015 PT:= int_to_sec(tm_strt_mtr_ignr_vlv));
0016 ELSE
0017 tmr_ignr_vlv_mode(
0018 IN:= FALSE,
0019 PT:= int_to_sec(tm_strt_mtr_ignr_vlv));
0020 END_IF
0021 END_IF
0022 (*Проверяем условие открытия заслонок и переходим к заданию частоты*)
0023 IF di_vlv_op OR tmr_ignr_vlv_mode.Q THEN
0024 sys_state := START_LNCH_FAN;
0025 END_IF
0026
0027 (*Условие выхода из состояния*)
0028 IF NOT prmssn_sup OR trg_stop_sup.Q THEN
0029 sys_state := STOPPING;
0030 END_IF
case_run_reg (ST)
0001 IF NOT prmssn_sup OR trg_stop_sup.Q THEN
0002 sys_state := STOPPING;
0003 END_IF
case_stopping (ST)
0001 fan.msg_lnch := FALSE;
0002 fan_r.msg_lnch := FALSE;
0003
0004 IF NOT di_vlv_op THEN
0005 sys_state := OFF;
0006 END_IF
case_strt_heat (ST)
0001 tmr_sup_strt.in := TRUE;
0002
0003 IF htr_invrs_temp_sp < htr_invrs_temp THEN
0004 sys_state := START_OP_VLV;
0005 ELSIF tmr_sup_strt.Q AND NOT Htr.wntr_ready THEN
0006 alm_preheat := TRUE;
0007 sys_state := STOPPING;
0008 END_IF
0009
0010 IF NOT prmssn_sup OR trg_stop_sup.Q THEN
0011 sys_state := STOPPING;
0012 END_IF
0013
freq_reg (ST)
0001 IF y.pos_open OR tmr_ignr_vlv_mode.Q THEN
0002 IF di_air_dirt_2 OR di_air_dirt_1 THEN
0003 alm_freq_ctrl := TRUE;
0004 END_IF
0005
0006 IF di_air_dirt_2 AND NOT di_air_dirt_1 THEN
0007 warn_snsr_freq := TRUE;

```

```

0008 ELSE
0009 warn_snsr_freq := FALSE;
0010 END_IF
0011
0012 IF di_air_dirt_1 THEN
0013 tmr_freq_reg_lvl_1.IN := TRUE;
0014 IF di_air_dirt_2 THEN
0015 tmr_freq_reg_lvl_2.IN := TRUE;
0016 ELSE
0017 tmr_freq_reg_lvl_2.IN := FALSE;
0018 END_IF
0019 ELSE
0020 tmr_freq_reg_lvl_1.IN := FALSE;
0021 END_IF
0022
0023 tmr_freq_reg_lvl_1(
0024 PT:= int_to_sec(tm_freq_reg),
0025 Q=> man_freq_reg_lvl_1,
0026 );
0027
0028 tmr_freq_reg_lvl_2(
0029 PT:= int_to_sec(tm_freq_reg),
0030 Q=> man_freq_reg_lvl_2,
0031 );
0032 IF man_freq_reg_lvl_1 AND NOT man_freq_reg_lvl_2 THEN
0033 spd_p := REAL_TO_WORD(freq_lvl_1_po1_1);
0034 spd_p_r := REAL_TO_WORD(freq_lvl_1_po1_1r);
0035 ELSIF man_freq_reg_lvl_1 AND man_freq_reg_lvl_2 THEN
0036 spd_p := REAL_TO_WORD(freq_lvl_2_po1_1);
0037 spd_p_r := REAL_TO_WORD(freq_lvl_2_po1_1r);
0038 ELSE
0039 spd_p := REAL_TO_WORD(freq_po1_1r);
0040 spd_p_r := REAL_TO_WORD(freq_po1_1r);
0041 END_IF
0042
0043 IF man_freq_reg_lvl_1 AND NOT man_freq_reg_lvl_2 THEN
0044 cur_tm_freq_reg := sec_to_int(tmr_freq_reg_lvl_1.ET);
0045 ELSIF man_freq_reg_lvl_1 AND man_freq_reg_lvl_2 THEN
0046 cur_tm_freq_reg := sec_to_int(tmr_freq_reg_lvl_2.ET);
0047 ELSE
0048 cur_tm_freq_reg := 0;
0049 END_IF
0050 ELSE
0051 alrm_freq_ctrl := FALSE;
0052 spd_p := 0 ;
0053 spd_p_r := 0 ;
0054 END_IF
0055
0056
0057 freq_lim_p(LOW_FREQ_LIM:= LOW_FREQ_LIM, HIGH_FREQ_LIM:= HIGH_FREQ_LIM, freq_in:=
cur_spd_p);
0058 freq_lim_p_r(LOW_FREQ_LIM:= LOW_FREQ_LIM, HIGH_FREQ_LIM:= HIGH_FREQ_LIM,
freq_in:= cur_spd_p_r);
0059
0060
0061 IF do_strt_fan_p THEN
0062 cur_spd_p := spd_p;
0063 ELSE
0064 cur_spd_p := 0;
0065 END_IF
0066
0067 IF do_strt_fan_p_r THEN

```

```

0068 cur_spd_p_r := spd_p_r;
0069 ELSE
0070 cur_spd_p_r := 0;
0071 END_IF
0072
0073 ao_spd_p := freq_lim_p.freq_out;
0074 ao_spd_p_r := freq_lim_p_r.freq_out;
init_actors (ST)
0001 y(
0002 id := ID_SUP_VLV,
0003 rst_err := rst_err,
0004 tm_op_vlv := tm_op_vlv,
0005 cmd_open := fan.msg_lnch,
0006 pos_open := di_vlv_op,
0007 fire := fire,
0008 ignr_md := hmi_ignr_vlv_mode,
0009 err_arr := alrm_arr
0010 );
0011
0012 fan(
0013 id := ID_SUP_FAN,
0014 rst_err := rst_err,
0015 fire := fire,
0016 volt_check:= di_phase_ctrl,
0017 freq_err := FALSE,
0018 brkr := NOT di_alrm_fan_p,
0019 auto := di_auto_fan_p,
0020 sft_brkr := TRUE,
0021 run := (spd_p > 0) AND do_strt_fan_p,
0022 t_ctrl := TRUE,
0023 diff_ctrl := di_pds_p,
0024 tm_diff_err := tm_diff_err,
0025 err_arr := alrm_arr,
0026 state_vlv := y.pos_open,
0027 cmd_open =>
0028 );
0029
0030 fan_r(
0031 id := ID_SUP_FAN_R,
0032 rst_err := rst_err,
0033 fire := fire,
0034 volt_check:= di_phase_ctrl,
0035 freq_err := FALSE,
0036 brkr := NOT di_alrm_fan_p_r,
0037 auto := di_auto_fan_p_r,
0038 sft_brkr := TRUE,
0039 run := (spd_p_r > 0) AND do_strt_fan_p_r,
0040 t_ctrl := TRUE,
0041 diff_ctrl := di_pds_p,
0042 tm_diff_err := tm_diff_err,
0043 err_arr := alrm_arr,
0044 state_vlv := y.pos_open,
0045
0046 cmd_open =>
0047 );
init_sensors (ST)
0001 (*Приток*)
0002 temp_in(
0003 id := id_sup_in_temp,
0004 input := ai_out_temp,
0005 fire := fire,
0006 value => sup_in_temp,

```



```

0007 err_arr := alm_arr,
0008 rst_err := rst_err
0009
0010 );
0011
0012 (*Hapyж*)
0013 temp_out(
0014 id := id_sup_out_temp,
0015 input := ai_sup_temp,
0016 use_defval := TRUE,
0017 fire := fire,
0018 value => sup_out_temp,
0019 err_arr := alm_arr,
0020 rst_err := rst_err
0021 );
oprtn_time (ST)
0001 oprtn_p(
0002 t_q := di_pds_p AND do_strt_fan_p,
0003 t_h1_work := tm_oprtn_p,
0004 );
0005
0006 oprtn_p_r(
0007 t_q := di_pds_p AND do_strt_fan_p_r,
0008 t_h1_work := tm_oprtn_p_r
0009 );
rotation (ST)
0001 IF strt_sup AND NOT (sys_state = START_HEATING) THEN
0002 rttn(
0003 alm_mtr_1 := main_sup_fan_alm,
0004 alm_mtr_2 := main_sup_fan_r_alm,
0005 man_strt_mtr_1 := FALSE,
0006 man_strt_mtr_2 := FALSE,
0007 mtr_enbl_1 := fan.auto AND fan.msg_lnch,
0008 mtr_enbl_2 := fan_r.auto AND fan_r.msg_lnch,
0009 sp_tm_rttn := tm_rttn_fan,
0010 tm_dl_rttn := tm_dl_rttn_fan,
0011 strt_mtr_1 => do_strt_fan_p,
0012 strt_mtr_2 => do_strt_fan_p_r
0013 );
0014 ELSE
0015 do_strt_fan_p := FALSE;
0016 do_strt_fan_p_r := FALSE;
0017 END_IF
Exhs (PRG-ST)
0001 PROGRAM Exhs
0002 VAR_INPUT
0003 rst_err: BOOL ;
0004 END_VAR
0005 VAR
0006 exh_1 : Motor;
0007 exh_2 : Motor;
0008 exh_3 : Motor;
0009 oprtn_exh_1 : Operating; (*Hаpaбoтka*)
0010
0011 oprtn_exh_2 : Operating; (*Hаpaбoтka*)
0012
0013 oprtn_exh_3 : Operating; (*Hаpaбoтka*)
0014
0015 END_VAR
0001 init;
0002 alm_gen;
0003 oprtn_time;

```

```

0004
0005 IF scd_strt_exh_2 AND di_auto_fan_exh_2 THEN
0006 exh_1.msg_lnch := TRUE;
0007 ELSE
0008 exh_1.msg_lnch := FALSE;
0009 END_IF
0010
0011 IF scd_strt_exh_3 AND di_auto_fan_exh_3 THEN
0012 exh_2.msg_lnch := TRUE;
0013 ELSE
0014 exh_2.msg_lnch := FALSE;
0015 END_IF
0016
0017 IF scd_strt_exh_1 AND di_auto_fan_exh_1 THEN
0018 exh_3.msg_lnch := TRUE;
0019 ELSE
0020 exh_3.msg_lnch := FALSE;
0021 END_IF
alm_gen (ST)
0001 IF NOT fire THEN
0002 IF alm_exh_exh_1 OR NOT di_auto_fan_exh_1 THEN
0003 scd_stop_exh_1 := TRUE;
0004 ELSE
0005 scd_stop_exh_1 := FALSE;
0006 END_IF
0007
0008 IF alm_exh_exh_2 OR NOT di_auto_fan_exh_2 THEN
0009 scd_stop_exh_2 := TRUE;
0010 ELSE
0011 scd_stop_exh_2 := FALSE;
0012 END_IF
0013
0014 IF alm_exh_exh_3 OR NOT di_auto_fan_exh_3 THEN
0015 scd_stop_exh_3 := TRUE;
0016 ELSE
0017 scd_stop_exh_3 := FALSE;
0018 END_IF
0019 ELSE
0020 scd_stop_exh_1 := TRUE;
0021 scd_stop_exh_2 := TRUE;
0022 scd_stop_exh_3 := TRUE;
0023 END_IF
init (ST)
0001 exh_1(
0002 id := ID_EXHS_FAN1,
0003 rst_err := rst_err,
0004 volt_check := di_phase_ctrl,
0005 brkr := NOT di_alm_fan_exh_1,
0006 auto := di_auto_fan_exh_1,
0007 run := di_run_fan_exh_1,
0008 diff_ctrl := TRUE,
0009 tm_diff_err := tm_diff_err,
0010 fire := fire,
0011 err_arr := alm_arr,
0012 state_vlv := TRUE,
0013 cmd_open => do_strt_fan_exh_1
0014 );
0015
0016 exh_2(
0017 id := ID_EXHS_FAN2,
0018 rst_err := rst_err,
0019 volt_check := di_phase_ctrl,

```

```

0020 brkr := NOT di_alm_fan_exh_2,
0021 auto := di_auto_fan_exh_2,
0022 run := di_run_fan_exh_2_1 OR di_run_fan_exh_2_2,
0023 diff_ctrl := di_pds_exh_2,
0024 tm_diff_err := tm_diff_err,
0025 fire := fire,
0026 err_arr := alm_arr,
0027 state_vlv := TRUE,
0028 cmd_open => do_strt_fan_exh_2
0029 );
0030
0031 exh_3(
0032 id := ID_EXHS_FAN3,
0033 rst_err := rst_err,
0034 volt_check := di_phase_ctrl,
0035 brkr := NOT di_alm_fan_exh_3,
0036 auto := di_auto_fan_exh_3,
0037 run := di_run_fan_exh_3,
0038 diff_ctrl := TRUE,
0039 tm_diff_err := tm_diff_err,
0040 fire := fire,
0041 err_arr := alm_arr,
0042 state_vlv := TRUE,
0043 cmd_open => do_strt_fan_exh_3
0044 );
oprtn_time (ST)
0001 oprtn_exh_1(
0002 t_q := di_run_fan_exh_1,
0003 t_h1_work := tm_oprtn_exh_1
0004 );
0005
0006 oprtn_exh_2(
0007 t_q := di_run_fan_exh_2_1 OR di_run_fan_exh_2_2,
0008 t_h1_work := tm_oprtn_exh_2
0009 );
0010
0011 oprtn_exh_3(
0012 t_q := di_run_fan_exh_3,
0013 t_h1_work := tm_oprtn_exh_3
0014 );
_README (PRG-ST)
0001 PROGRAM _README
0002 VAR
0003 END_VAR
0001 (*)
0002 Программа для контроллеров:
0003 IP ADDRESS: 10.0.6.10
0004 *)
HMI (PRG-ST)
0001 PROGRAM HMI
0002 VAR
0003 hmi_var_processor : hmi_in_out;
0004 arr_in_out_coil : ARRAY [1..MAX_NMBR_ITEMS] OF BOOL;
0005 arr_in_out_reg : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0006 arr_sf_in_out_coil : ARRAY [1..MAX_NMBR_ITEMS] OF BOOL;
0007 arr_sf_in_out_reg : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0008 arr_r_state : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0009 arr_w_state : ARRAY [1..MAX_NMBR_ITEMS] OF WORD;
0010 END_VAR
0001 conf_arr;
0002 conf_rw;
0003 conf_w;

```

```
conf_arr (ST)
0001 (*Объявление несохраняемых переменных*)
0002 arr_in_out_coil[1] := hmi_scd_strt_pv_p;
0003 arr_in_out_coil[2] := hmi_scd_strt_pv_v;
0004 arr_in_out_coil[3] := hmi_scd_strt_exh_1;
0005 arr_in_out_coil[4] := hmi_scd_strt_exh_2;
0006 arr_in_out_coil[5] := hmi_scd_strt_exh_3;
0007 arr_in_out_coil[6] := hmi_set_strt_mode;
0008 arr_in_out_coil[7] := hmi_winter_mode;
0009 arr_in_out_coil[8] := hmi_winter_mode_auto;
0010 arr_in_out_coil[9] := hmi_ignr_vlv_mode;
0011
0012 arr_in_out_reg[1] := hmi_sup_out_temp_sp;
0013 arr_in_out_reg[2] := hmi_freq_po1_1;
0014 arr_in_out_reg[3] := hmi_freq_po1_1r;
0015 arr_in_out_reg[4] := hmi_htr_vlv_kp;
0016 arr_in_out_reg[5] := hmi_htr_vlv_tn;
0017 arr_in_out_reg[6] := hmi_htr_vlv_tv;
0018 arr_in_out_reg[7] := hmi_tm_op_vlv;
0019 arr_in_out_reg[8] := hmi_tm_strt_mtr_ignr_vlv;
0020 arr_in_out_reg[9] := hmi_tm_sup_strt;
0021 arr_in_out_reg[10] := hmi_tm_diff_err;
0022 arr_in_out_reg[11] := hmi_tm_strt_mtr;
0023 arr_in_out_reg[12] := hmi_htr_invrs_temp_err_sp;
0024 arr_in_out_reg[13] := hmi_sup_out_temp_err;
0025
0026 (*Объявление сохраняемых переменных*)
0027 arr_sf_in_out_coil[1] := sf_hmi_scd_strt_pv_p;
0028 arr_sf_in_out_coil[2] := sf_hmi_scd_strt_pv_v;
0029 arr_sf_in_out_coil[3] := sf_hmi_scd_strt_exh_1;
0030 arr_sf_in_out_coil[4] := sf_hmi_scd_strt_exh_2;
0031 arr_sf_in_out_coil[5] := sf_hmi_scd_strt_exh_3;
0032 arr_sf_in_out_coil[6] := sf_hmi_set_strt_mode;
0033 arr_sf_in_out_coil[7] := sf_hmi_winter_mode;
0034 arr_sf_in_out_coil[8] := sf_hmi_winter_mode_auto;
0035 arr_sf_in_out_coil[9] := sf_hmi_ignr_vlv_mode;
0036
0037 arr_sf_in_out_reg[1] := sf_hmi_sup_out_temp_sp;
0038 arr_sf_in_out_reg[2] := sf_hmi_freq_po1_1;
0039 arr_sf_in_out_reg[3] := sf_hmi_freq_po1_1r;
0040 arr_sf_in_out_reg[4] := htr_vlv_kp;
0041 arr_sf_in_out_reg[5] := htr_vlv_tn;
0042 arr_sf_in_out_reg[6] := htr_vlv_tv;
0043 arr_sf_in_out_reg[7] := tm_op_vlv;
0044 arr_sf_in_out_reg[8] := tm_strt_mtr_ignr_vlv;
0045 arr_sf_in_out_reg[9] := tm_sup_strt;
0046 arr_sf_in_out_reg[10] := tm_diff_err;
0047 arr_sf_in_out_reg[11] := tm_strt_mtr;
0048 arr_sf_in_out_reg[12] := htr_invrs_temp_err_sp;
0049 arr_sf_in_out_reg[13] := sup_out_temp_err;
0050
0051 (*Объявление состояний*)
0052 arr_r_state[1] := h_iw1_state;
0053 arr_r_state[2] := h_iw2_state;
0054 arr_r_state[3] := h_iw3_state;
0055 arr_r_state[4] := h_iw4_state;
0056 arr_r_state[5] := h_iw5_state;
0057 arr_r_state[6] := h_iw6_state;
0058 arr_r_state[7] := h_iw7_state;
0059 arr_r_state[8] := h_iw8_state;
0060 arr_r_state[9] := h_iw9_state;
0061 arr_r_state[10] := h_iw10_state;
```

```

0062 arr_r_state[11] := h_iw11_state;
0063 arr_r_state[12] := h_iw12_state;
0064 arr_r_state[13] := h_iw13_state;
0065 arr_r_state[14] := h_iw14_state;
0066
0067 arr_w_state[1] := h_qix1_state;
0068 arr_w_state[2] := h_qix2_state;
0069 arr_w_state[3] := h_qiw2_state;
0070 arr_w_state[4] := h_qiw3_state;
0071 arr_w_state[5] := h_qiw4_state;
0072 arr_w_state[6] := h_qiw5_state;
0073 arr_w_state[7] := h_qiw6_state;
0074 arr_w_state[8] := h_qiw7_state;
0075 arr_w_state[9] := h_qiw8_state;
0076 arr_w_state[10] := h_qiw9_state;
0077 arr_w_state[11] := h_qiw10_state;
0078 arr_w_state[12] := h_qiw11_state;
0079 arr_w_state[13] := h_qiw12_state;
0080 arr_w_state[14] := h_qiw13_state;
0081 arr_w_state[15] := h_qiw14_state;
conf_rw (ST)
0001 (*Чтение из панели*)
0002 hmi_scd_strt_pv_p := h_iw1.0 ; (* Регистр 180 *)
0003 hmi_scd_strt_pv_v := h_iw1.1 ;
0004 hmi_scd_strt_exh_1 := h_iw1.2 ;
0005 hmi_scd_strt_exh_2 := h_iw1.3 ;
0006 hmi_scd_strt_exh_3 := h_iw1.4 ;
0007 hmi_set_strt_mode := h_iw1.5 ;
0008 hmi_winter_mode := h_iw1.6 ;
0009 hmi_winter_mode_auto := h_iw1.7 ;
0010 hmi_ignr_vlv_mode := h_iw1.8 ; (* Регистр 188 *)
0011
0012 hmi_sup_out_temp_sp := h_iw2 ; (* Регистр 51 *)
0013 hmi_freq_po1_1 := h_iw3 ; (* Регистр 52 *)
0014 hmi_freq_po1_1r := h_iw3 ;
0015 hmi_htr_vlv_kp := h_iw5 ;
0016 hmi_htr_vlv_tn := h_iw6 ;
0017 hmi_htr_vlv_tv := h_iw7 ;
0018 hmi_tm_op_vlv := h_iw8 ;
0019 hmi_tm_strt_mtr_ignr_vlv := h_iw9 ;
0020 hmi_tm_sup_strt := h_iw10 ;
0021 hmi_tm_diff_err := h_iw11 ;
0022 hmi_tm_strt_mtr := h_iw12 ;
0023 hmi_htr_invrs_temp_err_sp := h_iw13 ;
0024 hmi_sup_out_temp_err := h_iw14 ;
0025 (*hmi_tm_rttm_pmp := h_iw13;
0026 hmi_tm_rttm_fan := h_iw14;*)
0027
0028 (*Запись в панель*)
0029 h_qix1.0 := sf_hmi_scd_strt_pv_p ;
0030 h_qix1.1 := sf_hmi_scd_strt_pv_v ;
0031 h_qix1.2 := sf_hmi_scd_strt_exh_1 ;
0032 h_qix1.3 := sf_hmi_scd_strt_exh_2 ;
0033 h_qix1.4 := sf_hmi_scd_strt_exh_3 ;
0034 h_qix1.5 := sf_hmi_set_strt_mode ;
0035 h_qix1.6 := sf_hmi_winter_mode ;
0036 h_qix1.7 := sf_hmi_winter_mode_auto ;
0037 h_qix2.0 := sf_hmi_ignr_vlv_mode ;
0038
0039 h_qiw2 := sf_hmi_sup_out_temp_sp ; (* Регистр 51 *)
0040 h_qiw3 := sf_hmi_freq_po1_1 ; (* Регистр 52 *)
0041 h_qiw4 := sf_hmi_freq_po1_1r ; (* Регистр 53 *)

```

```

0042 h_qiw5 := htr_vlv_kp ; (* Регистр 54 *)
0043 h_qiw6 := htr_vlv_tn ; (* Регистр 55 *)
0044 h_qiw7 := htr_vlv_tv ; (* Регистр 56 *)
0045 h_qiw8 := tm_op_vlv ; (* Регистр 57 *)
0046 h_qiw9 := tm_strt_mtr_ignr_vlv ; (* Регистр 58 *)
0047 h_qiw10 := tm_sup_strt ; (* Регистр 59 *)
0048 h_qiw11 := tm_diff_err ; (* Регистр 60 *)
0049 h_qiw12 := tm_strt_mtr ; (* Регистр 61 *)
0050 h_qiw13 := htr_invrs_temp_err_sp ; (* Регистр 61 *)
0051 h_qiw14 := sup_out_temp_err ; (* Регистр 61 *)
0052 (*h_qiw13 := tm_rttn_pmp ; (* Регистр 62 *)
0053 h_qiw14 := tm_rttn_fan ; (* Регистр 63 *)*)
0054
0055 (*Обработка разрешений на чтение и запись*)
0056 hmi_var_processor(
0057 arr_in_out_coil := arr_in_out_coil,
0058 arr_sf_in_out_coil := arr_sf_in_out_coil,
0059 arr_in_out_reg := arr_in_out_reg,
0060 arr_sf_in_out_reg := arr_sf_in_out_reg,
0061 check_by_var := arr_in_out_reg[1],
0062 arr_r_state => arr_r_state ,
0063 arr_w_state => arr_w_state
0064 );
conf_w (ST)
0001 (*Запись в панель*)
0002 qx1.0 := alrm_sup_fan_diff ; (* Регистр 200 *)
0003 qx1.1 := alrm_sup_fan_brkr ;
0004 qx1.2 := alrm_sup_fan_strt ;
0005 qx1.3 := alrm_sup_fan_r_diff ;
0006 qx1.4 := alrm_sup_fan_r_brkr ;
0007 qx1.5 := alrm_sup_fan_r_strt ;
0008 qx1.6 := warn_snsr_freq ;
0009 qx1.7 := alrm_cnt_freq
0010
0011 qx2.0 := alrm_cnt_pmp ; (* Регистр 208 *)
0012 qx2.1 := alrm_sup_vlv ;
0013 qx2.2 := alrm_sys ;
0014 qx2.3 := alrm_invrs_temp ;
0015 qx2.4 := alrm_preheat ;
0016 qx2.5 := alrm_ts ;
0017 qx2.6 := alrm_cnt_preheat ;
0018 qx2.7 := alrm_sens_in ;
0019 *Приток*)
0020 qx3.0 := alrm_sens_out ; (* Регистр 216 *)
0021
0022 qx3.1 := alrm_sens_invrt ;
0023 qx3.2 := (*warn_mismatch*) FALSE ;
0024 qx3.3 := warn_fltr_sup ;
0025 qx3.4 := warn_not_auto ;
0026 qx3.5 := alrm_htr_pmp ;
0027 qx3.6 := alrm_htr_pmp_r ;
0028 qx3.7 := alrm_exh_exh_1 ;
0029 qx4.0 := alrm_exh_exh_2 ; (* Регистр 224 *)
0030
0031 qx4.1 := alrm_exh_exh_3 ;
0032 qx4.2 := alrm_sup_temp ;
0033 qx4.3 := alrm_brk_sens_invrt ;
0034 qx4.4 := alrm_brk_sens_in ;
0035 qx4.5 := alrm_brk_sens_out ;
0036 qx4.6 := alrm_brk_temp_rm_1 ;
0037 qx4.7 := alrm_brk_temp_rm_2 ;
0038

```

```

0039 qx7.0 := FALSE ;
0040 * Регистр 248 *)
0041 qx7.1 := FALSE ;
0042 qx7.2 := FALSE ;
0043 qx7.3 := FALSE ;
0044 qx7.4 := FALSE ;
0045 qx7.5 := FALSE ;
0046 qx7.6 := FALSE ;
0047 qx7.7 := FALSE ;
0048
0049 qx6.0 := do_strt_fan_p ; (* Регистр 240 *)
0050 qx6.1 := do_strt_fan_p_r ;
0051 qx6.2 := mb_do_strt_v01_1 ;
0052 qx6.3 := mb_do_strt_v01_1r ;
0053 qx6.4 := di_run_fan_exh_1 ;
0054 qx6.5 := di_run_fan_exh_2_1 ;
0055 qx6.6 := di_run_fan_exh_3 ;
0056 qx6.7 := di_run_pump_p ;
0057
0058 qx8.0 := di_run_pump_p_r ; (* Регистр 256 *)
0059 qx8.1 := di_pds_p ;
0060 qx8.2 := mb_di_pds_fan_v01_1 ;
0061 qx8.3 := di_pds_exh_2 ;
0062 qx8.4 := di_pds_exh_3 ;
0063 qx8.5 := hmi_winter_mode_auto ;
0064 qx8.6 := winter_mode ;
0065 qx8.7 := cnnctn_2 ;
0066
0067 qw1 := REAL_TO_INT(htr_invrs_temp) ; (* Регистр 100 *)
0068 qw2 := REAL_TO_INT(sup_out_temp) ; (* Регистр 101 *)
0069 qw3 := REAL_TO_INT(sup_in_temp) ; (* Регистр 102 *)
0070 qw4 := REAL_TO_INT(ai_exh_temp_1) ; (* Регистр 103 *)
0071 qw5 := REAL_TO_INT(ai_exh_temp_2) ; (* Регистр 104 *)
0072 qw6 := cur_spd_p ; (* Регистр 105 *)
0073 qw7 := cur_spd_p_r ; (* Регистр 106 *)
0074 qw8 := DWORD_TO_WORD(tm_oprtn_p) ; (* Регистр 107 *)
0075 qw9 := DWORD_TO_WORD(tm_oprtn_p_r) ; (* Регистр 108 *)
0076 qw10 := DWORD_TO_WORD(tm_oprtn_pmp) ; (* Регистр 109 *)
0077 qw11 := DWORD_TO_WORD(tm_oprtn_pmp_r) ; (* Регистр 110 *)
0078 qw12 := DWORD_TO_WORD(tm_oprtn_exh_2) ; (* Регистр 111 *)
0079 qw13 := DWORD_TO_WORD(tm_oprtn_exh_3) ; (* Регистр 112 *)
0080 qw14 := DWORD_TO_WORD(tm_oprtn_exh_1) ; (* Регистр 113 *)
0081 qw15 := ao_y1/10 ; (* Регистр 114 *)
0082 qw16 := reg_state ; (* Регистр 115 *)
0083 qw17 := sys_state ; (* Регистр 116 *)
0084 qw18 := mb_cur_spd_v01_1 ; (* Регистр 117 *)
0085 qw19 := mb_cur_spd_v01_1p ; (* Регистр 118 *)
0086 qw20 := REAL_TO_INT(htr_invrs_temp_sp); (* Регистр 119 *)
Sys (PRG-ST)
0001 PROGRAM Sys
0002 VAR
0002 VAR
0003 brk_sens_1 : check_break;
0004 brk_sens_2 : check_break;
0005 brk_sens_3 : check_break;
0006 brk_sens_4 : check_break;
0007 brk_sens_5 : check_break;
0008 brk_sens_6 : check_break;
0009 brk_sens_7 : check_break;
0010 brk_sens_8 : check_break;
0011 wntr_hyst : HYSTERESIS;
0012 wntr_hyst_low : REAL := 12;

```





```

0024 scd_strt_exh_1 := FALSE ;
0025 END_IF
0026
0027 IF NOT scd_stop_exh_2 THEN
0028 scd_strt_exh_2 := hmi_scd_strt_exh_2 ;
0029 ELSE
0030 scd_strt_exh_2 := FALSE ;
0031 END_IF
0032
0033 IF NOT scd_stop_exh_3 THEN
0034 scd_strt_exh_3 := hmi_scd_strt_exh_3 ;
0035 ELSE
0036 scd_strt_exh_3 := FALSE ;
0037 END_IF
mode (ST)
0001 wntr_hyst(
0002 IN := REAL_TO_INT(sup_in_temp * 10),
0003 HIGH := REAL_TO_INT(wntr_hyst_high * 10),
0004 LOW := REAL_TO_INT(wntr_hyst_low * 10)
0005 );
0006
0007 IF hmi_winter_mode_auto THEN
0008 winter_mode := wntr_hyst.OUT;
0009 ELSE
0010 winter_mode := hmi_winter_mode;
0011 END_IF
plc (ST)
0001 (*Запись в plc*)
0002 plc_qx1.0 := di_alarm_rst ; (* Coil 0 *) (* Регистр 0 *)
0003 plc_qx1.1 := di_fire ; (* Coil 1 *)
0004 plc_qx1.2 := di_phase_ctrl ; (* Coil 2 *)
0005 plc_qx1.3 := di_air_dirt_1 ; (* Coil 3 *)
0006 plc_qx1.4 := di_air_dirt_2 ; (* Coil 4 *)
0007 plc_qx1.5 := di_ts ; (* Coil 5 *)
0008 plc_qx1.6 := di_auto_fan_p ; (* Coil 6 *)
0009 plc_qx1.7 := di_alarm_fan_p ; (* Coil 7 *)
0010 plc_qx2.0 := di_auto_pump_p ; (* Coil 8 *)
0011 plc_qx2.1 := di_run_pump_p ; (* Coil 9 *)
0012 plc_qx2.2 := di_alarm_pump_p ; (* Coil 10 *)
0013 plc_qx2.3 := di_vlv_op ; (* Coil 11 *)
0014 plc_qx2.4 := di_pds_fltr ; (* Coil 12 *)
0015 plc_qx2.5 := di_pds_p ; (* Coil 13 *)
0016 plc_qx2.6 := di_pds_exh_3 ; (* Coil 14 *)
0017 plc_qx2.7 := di_pds_exh_2 ; (* Coil 15 *)
0018 plc_qx3.0 := di_auto_fan_p_r ; (* Coil 16 *) (* Регистр 1 *)
0019 plc_qx3.1 := di_alarm_fan_p_r ; (* Coil 17 *)
0020 plc_qx3.2 := di_auto_fan_exh_1 ; (* Coil 18 *)
0021 plc_qx3.3 := di_run_fan_exh_1 ; (* Coil 19 *)
0022 plc_qx3.4 := di_alarm_fan_exh_1 ; (* Coil 20 *)
0023 plc_qx3.5 := di_auto_fan_exh_2 ; (* Coil 21 *)
0024 plc_qx3.6 := di_run_fan_exh_2_1 ; (* Coil 22 *)
0025 plc_qx3.7 := di_alarm_fan_exh_2 ; (* Coil 23 *)
0026 plc_qx4.0 := di_auto_fan_exh_3 ; (* Coil 24 *)
0027 plc_qx4.1 := di_run_fan_exh_3 ; (* Coil 25 *)
0028 plc_qx4.2 := di_alarm_fan_exh_3 ; (* Coil 26 *)
0029 plc_qx4.3 := di_run_fan_exh_2_2 ; (* Coil 27 *)
0030 plc_qx4.4 := di_run_pump_p_r ; (* Coil 28 *)
0031 plc_qx4.5 := di_alarm_pump_p_r ; (* Coil 29 *)
0032 plc_qx4.6 := di_auto_pump_p_r ; (* Coil 30 *)
0033 plc_qx4.7 := do_strt_pump_p ; (* Coil 31 *)
0034 plc_qx5.0 := do_strt_pump_pr ; (* Coil 32 *) (* Регистр 2 *)
0035 plc_qx5.1 := do_strt_fan_exh_1 ; (* Coil 33 *)

```

```
0036 plc_qx5.2 := do_strt_fan_p ; (* Coil 34 *)
0037 plc_qx5.3 := do_strt_fan_p_r ; (* Coil 35 *)
0038 plc_qx5.4 := do_main_alarm ; (* Coil 36 *)
0039 plc_qx5.5 := do_strt_fan_exh_2 ; (* Coil 37 *)
0040 plc_qx5.6 := do_strt_fan_exh_3 ; (* Coil 38 *)
0041 plc_qx5.7 := alarm_sup_fan_diff ; (* Coil 39 *)
0042 plc_qx6.0 := alarm_sup_fan_brkr ; (* Coil 40 *)
0043 plc_qx6.1 := alarm_sup_fan_strt ; (* Coil 41 *)
0044 plc_qx6.2 := alarm_sup_fan_r_diff ; (* Coil 42 *)
0045 plc_qx6.3 := alarm_sup_fan_r_brkr ; (* Coil 43 *)
0046 plc_qx6.4 := alarm_sup_fan_r_strt ; (* Coil 44 *)
0047 plc_qx6.5 := alarm_sup_vlv ; (* Coil 45 *)
0048 plc_qx6.6 := FALSE ; (* Coil 46 *)
0049 plc_qx6.7 := alarm_sens_in ; (* Coil 47 *)
0050 plc_qx7.0 := alarm_sens_out ; (* Coil 48 *) (* Регистр 3 *)
0051 plc_qx7.1 := alarm_sens_invrt ; (* Coil 49 *)
0052 plc_qx7.2 := FALSE ; (* Coil 50 *)
0053 plc_qx7.3 := FALSE ; (* Coil 51 *)
0054 plc_qx7.4 := FALSE ; (* Coil 52 *)
0055 plc_qx7.5 := FALSE ; (* Coil 53 *)
0056 plc_qx7.6 := alarm_htr_pmp ; (* Coil 54 *)
0057 plc_qx7.7 := alarm_htr_pmp_r ; (* Coil 55 *)
0058 plc_qx8.0 := FALSE ; (* Coil 56 *)
0059 plc_qx8.1 := alarm_sys ; (* Coil 57 *)
0060 plc_qx8.2 := alarm_invrs_temp ; (* Coil 58 *)
0061 plc_qx8.3 := alarm_preheat ; (* Coil 59 *)
0062 plc_qx8.4 := alarm_exh_exh_2 ; (* Coil 60 *)
0063 plc_qx8.5 := alarm_exh_exh_3 ; (* Coil 61 *)
0064 plc_qx8.6 := alarm_exh_exh_1 ; (* Coil 62 *)
0065 plc_qx8.7 := alarm_ts ; (* Coil 63 *)
0066 plc_qx9.0 := alarm_cnt_preheat ; (* Coil 64 *) (* Регистр 4 *)
0067 plc_qx9.1 := warn_mismatch ; (* Coil 65 *)
0068 plc_qx9.2 := warn_snsr_freq ; (* Coil 66 *)
0069 plc_qx9.3 := warn_fltr_sup ; (* Coil 67 *)
0070 plc_qx9.4 := warn_not_auto ; (* Coil 68 *)
0071 plc_qx9.5 := FALSE ; (* Coil 69 *)
0072 plc_qx9.6 := main_alarm ; (* Coil 70 *)
0073 plc_qx9.7 := main_sup_alarm ; (* Coil 71 *)
0074 plc_qx10.0 := FALSE ; (* Coil 72 *)
0075 plc_qx10.1 := main_htr_alarm ; (* Coil 73 *)
0076 plc_qx10.2 := main_exhs_alarm ; (* Coil 74 *)
0077 plc_qx10.3 := (*main_warn*) alarm_freq_ctrl ; (* Coil 75 *)
0078 plc_qx10.4 := hmi_set_strt_mode ; (* Coil 76 *)
0079 plc_qx10.5 := winter_mode ; (* Coil 77 *)
0080 plc_qx10.6 := hmi_scd_strt_pv_p ; (* Coil 78 *)
0081 plc_qx10.7 := strt_exh ; (* Coil 79 *)
0082 plc_qd1 := ai_invr_temp ; (* Регистр 30 *)
0083 plc_qd2 := ai_sup_temp ; (* Регистр 32 *)
0084 plc_qd3 := ai_state_y1 ; (* Регистр 34 *)
0085 plc_qd4 := ai_exh_temp_1 ; (* Регистр 36 *)
0086 plc_qd5 := ai_exh_temp_2 ; (* Регистр 38 *)
0087 plc_qd6 := ai_out_temp ; (* Регистр 40 *)
0088 plc_qd7 := ao_y1 ; (* Регистр 42 *)
0089 plc_qd8 := cur_spd_p ; (* Регистр 44 *)
0090 plc_qd9 := cur_spd_p_r ; (* Регистр 46 *)
0091 plc_qd10 := cur_tm_freq_reg ; (* Регистр 48 *)
0092 plc_qd11 := spd_p ; (* Регистр 50 *)
0093 plc_qd12 := spd_p_r ; (* Регистр 52 *)
0094
0095 (*Чтение из plc*)
0096 mb_di_alarm_rst := plc_qx13.0 ; (* Coil 96 *) (* Регистр 5 *)
0097 mb_di_fire := plc_qx13.1 ; (* Coil 97 *)
```

```

0098 mb_di_phase_ctrl := plc_qx13.2 ; (* Coil 98 *)
0099 mb_di_auto_fan_v01_1 := plc_qx13.3 ; (* Coil 99 *)
0100 mb_di_alarm_fan_v01_1 := plc_qx13.4 ; (* Coil 100 *)
0101 mb_di_exh_vlv_open := plc_qx13.5 ; (* Coil 101 *)
0102 mb_di_pds_fltr := plc_qx13.6 ; (* Coil 102 *)
0103 mb_di_pds_fan_v01_1 := plc_qx13.7 ; (* Coil 103 *)
0104 mb_di_pds_fan_v1_3 := plc_qx14.0 ; (* Coil 104 *)
0105 mb_di_auto_fan_v01_1r := plc_qx14.1 ; (* Coil 105 *)
0106 mb_di_alarm_fan_v01_1r := plc_qx14.2 ; (* Coil 106 *)
0107 mb_di_auto_fan_v1_3 := plc_qx14.3 ; (* Coil 107 *)
0108 mb_di_run_fan_v1_3 := plc_qx14.4 ; (* Coil 108 *)
0109 mb_di_alarm_fan_v1_3 := plc_qx14.5 ; (* Coil 109 *)
0110 mb_do_strt_v01_1 := plc_qx14.6 ; (* Coil 110 *)
0111 mb_do_strt_v01_1r := plc_qx14.7 ; (* Coil 111 *)
0112
0113 mb_do_main_alarm := plc_qx15.0 ; (* Coil 112 *) (* Регистр 6 *)
0114 mb_do_strt_v1_3 := plc_qx15.1 ; (* Coil 113 *)
0115 mb_cnctn_2 := plc_qx15.2 ; (* Coil 114 *)
0116 mb_alarm_exh_fan_diff := plc_qx15.3 ; (* Coil 115 *)
0117 mb_alarm_exh_fan_brkr := plc_qx15.4 ; (* Coil 116 *)
0118 mb_alarm_exh_fan_strt := plc_qx15.5 ; (* Coil 117 *)
0119 mb_alarm_exh_fan_r_diff:= plc_qx15.6 ; (* Coil 118 *)
0120 mb_alarm_exh_fan_r_brkr:= plc_qx15.7 ; (* Coil 119 *)
0121 mb_alarm_exh_fan_r_strt:= plc_qx16.0 ; (* Coil 120 *)
0122 mb_alarm_exh_fan_r := plc_qx16.1 ; (* Coil 121 *)
0123 mb_alarm_exh_vlv := plc_qx16.2 ; (* Coil 122 *)
0124 mb_alarm_sys := plc_qx16.3 ; (* Coil 123 *)
0125 mb_alarm_exh_v1_3 := plc_qx16.4 ; (* Coil 124 *)
0126 mb_warn_fltr := plc_qx16.5 ; (* Coil 125 *)
0127 hmi_scd_strt_pv_v := plc_qx16.6 ; (* Coil 126 *)
0128 (*mb_scd_stop_pv_p := plc_qx16.7 ; (* Coil 127 *)*)
0129
0130 (*mb_scd_strt_pv_v := plc_qx17.0 ; (* Coil 128 *) (* Регистр 7 *)*)
0131 (*mb_scd_stop_pv_v := plc_qx17.1 ; (* Coil 129 *)*)
0132 (*mb_set_strt_mode := plc_qx17.2 ; (* Coil 130 *)*)
0133 (*mb_rst_err := plc_qx17.3 ; (* Coil 131 *)*)
0134 (*mb_winter_mode := plc_qx17.4 ; (* Coil 132 *)*)
0135 (*mb_winter_mode_auto := plc_qx17.5 ; (* Coil 133 *)*)
0136 (*hmi_scd_strt_pv_v := plc_qx17.6 ; (* Coil 133 *)*)
0137 mb_cur_spd_v01_1 := plc_qw1 ; (* Регистр 10 *)
0138 mb_cur_spd_v01_1p := plc_qw2 ; (* Регистр 11 *)
0139 mb_oprtng_v0_1 := plc_qw3 ; (* Регистр 12 *)
0140 mb_oprtng_v0_1p := plc_qw4 ; (* Регистр 13 *)
0141
reset_err (ST)
0001 fire := NOT di_fire;
0002 rst_err := di_alarm_rst;
0003 prtct_air := di_ts;
0004
0005 IF di_alarm_rst THEN
0006 Htr.rst_err := TRUE;
0007 Sup.rst_err := TRUE;
0008 Exhs.rst_err := TRUE;
0009 ELSE
0010 Htr.rst_err := FALSE;
0011 Sup.rst_err := FALSE;
0012 Exhs.rst_err := FALSE;
0013 END_IF
0014
0015 IF rst_err THEN
0016 alrm_cnt_preheat := FALSE;
0017 alrm_invrs_temp := FALSE;

```

```

0018 alrm_cnt_pmp := FALSE;
0019 alrm_preheat := FALSE;
0020 alrm_ts := FALSE;
0021 alrm_cnt_freq := FALSE;
0022 alrm_cnt_pmp := FALSE;
0023 alrm_sup_temp := FALSE;
0024 alrm_brk_dfb_vlv := FALSE;
0025 alrm_brk_sens_in := FALSE;
0026 alrm_brk_sens_out := FALSE;
0027 alrm_brk_sens_invrt := FALSE;
0028 alrm_brk_temp_rm_1 := FALSE;
0029 alrm_brk_temp_rm_2 := FALSE;
0030 alrm_sup_temp := FALSE;
0031 alrm_freq_ctrl := FALSE;
0032 END_IF
0033
0034
0035
set_err (ST)
0001 alrm_sys := NOT di_phase_ctrl;
0002 warn_fltr_sup := di_pds_fltr;
0003
0004
0005 brk_sens_1(sens_state:=ai_brkr_sens_1);
0006 brk_sens_2(sens_state:=ai_brkr_sens_2);
0007 brk_sens_3(sens_state:=ai_brkr_sens_3);
0008 brk_sens_4(sens_state:=ai_brkr_sens_4);
0009 brk_sens_5(sens_state:=ai_brkr_sens_5);
0010 brk_sens_6(sens_state:=ai_brkr_sens_6);
0011 brk_sens_7(sens_state:=ai_brkr_sens_7);
0012 brk_sens_8(sens_state:=ai_brkr_sens_8);
0013
0014 IF brk_sens_1.err THEN
0015 alrm_brk_sens_invrt := TRUE;
0016 END_IF
0017
0018 IF brk_sens_2.err THEN
0019 alrm_brk_sens_in := TRUE; (*Приток*)
0020 END_IF
0021
0022 IF brk_sens_4.err THEN
0023 alrm_brk_dfb_vlv := TRUE;
0024 END_IF
0025
0026 IF brk_sens_5.err THEN (*Электрощитова*)
0027 alrm_brk_temp_rm_1 := TRUE;
0028 END_IF
0029
0030 IF brk_sens_6.err THEN (*Сети связи*)
0031 alrm_brk_temp_rm_2 := TRUE;
0032 END_IF
0033
0034 IF brk_sens_7.err THEN
0035 alrm_brk_sens_out := TRUE; (*Наруж*)
0036 END_IF
0037
0038 alrm_sup_fan_diff := alrm_arr[ ID_SUP_FAN + 1 ];
0039 alrm_sup_fan_brkr := alrm_arr[ ID_SUP_FAN + 2 ];
0040 alrm_sup_fan_strt := alrm_arr[ ID_SUP_FAN + 7 ];
0041 alrm_sup_fan_r_diff := alrm_arr[ ID_SUP_FAN_R + 1 ];
0042 alrm_sup_fan_r_brkr := alrm_arr[ ID_SUP_FAN_R + 2 ];
0043 alrm_sup_fan_r_strt := alrm_arr[ ID_SUP_FAN_R + 7 ];

```

```

0044 alrm_sup_vlv := alrm_arr[ ID_SUP_VLV ];
0045 alrm_sens_in := alrm_arr[ ID_SUP_IN_TEMP ];
0046 alrm_sens_out := alrm_arr[ ID_SUP_OUT_TEMP ];
0047 alrm_sens_invrt := alrm_arr[ ID_HTR_INVRS_TEMP ];
0048 alrm_exh_exh_1 := alrm_arr[ ID_EXHS_FAN1 ];
0049 alrm_exh_exh_2 := alrm_arr[ ID_EXHS_FAN2 ];
0050 alrm_exh_exh_3 := alrm_arr[ ID_EXHS_FAN3 ];
0051 alrm_htr_pmp := alrm_arr[ ID_HTR_PMP ] AND winter_mode;
0052 alrm_htr_pmp_r := alrm_arr[ ID_HTR_PMP_R ] AND winter_mode;
0053
0054 main_sup_fan_alarm := (alrm_sup_fan_diff OR
0055 alrm_sup_fan_brkr OR
0056 alrm_sup_fan_strt);
0057
0058 main_sup_fan_r_alarm := (alrm_sup_fan_r_diff OR
0059 alrm_sup_fan_r_brkr OR
0060 alrm_sup_fan_r_strt);
0061
0062 main_sup_alarm := (alrm_sup_fan_diff OR
0063 alrm_sup_fan_brkr OR
0064 alrm_sup_fan_strt OR
0065 alrm_sup_fan_r_diff OR
0066 alrm_sup_fan_r_brkr OR
0067 alrm_sup_fan_r_strt OR
0068 alrm_sup_vlv OR
0069 alrm_sens_in OR
0070 alrm_sens_out OR
0071 alrm_sens_invrt);
0072
0073 main_htr_alarm := winter_mode AND (alrm_htr_pmp OR
0074 alrm_htr_pmp_r OR
0075 alrm_sys OR
0076 alrm_invrs_temp OR
0077 alrm_sup_temp OR
0078 alrm_preheat OR
0079 alrm_cnt_preheat OR
0080 alrm_ts);
0081
0082 main_exhs_alarm := (alrm_exh_exh_2 OR
0083 alrm_exh_exh_3 OR
0084 alrm_exh_exh_1);
0085
0086 main_alarm := (main_sup_alarm OR
0087 main_htr_alarm OR
0088 main_exhs_alarm);
0089
0090 main_warn := (warn_snsr_freq OR
0091 (warn_mismatch AND winter_mode) OR
0092 warn_fltr_sup OR
0093 (warn_not_auto AND winter_mode));
0094
0095 do_main_alarm := main_alarm (*OR main_warn*);
vrbls (ST)
0001 di_auto_fan_p_r := a1_1.0 ;
0002 di_alarm_fan_p_r := a1_1.2 ;
0003 di_auto_fan_exh_1 := a1_1.3 ;
0004 di_run_fan_exh_1 := a1_1.4 ;
0005 di_alarm_fan_exh_1 := a1_1.5 ;
0006 di_auto_fan_exh_2 := a1_1.6 ;
0007 di_run_fan_exh_2_1 := a1_1.7 ;
0008 di_alarm_fan_exh_2 := a1_1.8 ;
0009 di_auto_fan_exh_3 := a1_1.9 ;

```

```

0010 di_run_fan_exh_3 := a1_1.10 ;
0011 di_alrm_fan_exh_3 := a1_1.11 ;
0012 di_run_fan_exh_2_2 := a1_1.12 ;
0013 di_alrm_pump_p_r := a1_1.14 ;
0014
0015 di_run_pump_p_r := a1_1.13 ;
0016 di_auto_pump_p_r := a1_1.15 ;
ID_ALARM
0001 TYPE ID_ALARM :
0002 (
0003 ID_SUP_VLV := 0,
0004 ID_SUP_IN_TEMP := 1,
0005 ID_SUP_OUT_TEMP := 2,
0006 ID_HTR_INVRS_TEMP := 3,
0007 ID_SUP_FAN := 10,
0008 ID_SUP_FAN_R := 20,
0009
0010 ID_HTR_PMP := 30,
0011 ID_HTR_PMP_R := 40,
0012 ID_HTR_VLV := 50,
0013 ID_EXHS_FAN1 := 60,
0014 ID_EXHS_FAN2 := 70,
0015 ID_EXHS_FAN3 := 80
0016 );
0017 END_TYPE
REG_STATES
0001 TYPE REG_STATES :
0002 (
0003 NO_REG := 0, (* Не регулируется *)
0004 MAINTAIN := 1, (* Поддержание требуемой температуры при остановке системы зимой *)
0005 PREHEAT := 2, (* Предподогрев *)
0006 WAIT_REG := 3, (* Ожидание регулирования *)
0007 REGULATOR := 4 (* Алгоритм регулирования для калорифера и рекуператора *)
0008 );
0009 END_TYPE
SYS_STATES
0001 TYPE SYS_STATES :
0002 (
0003 OFF := 0 ,
0004 START_HEATING := 1 ,
0005 START_OP_VLV := 2 ,
0006 START_LNCH_FAN := 3 ,
0007 RUN_REG := 4 ,
0008 STOPPING := 5
0009 );
0010 END_TYPE

```