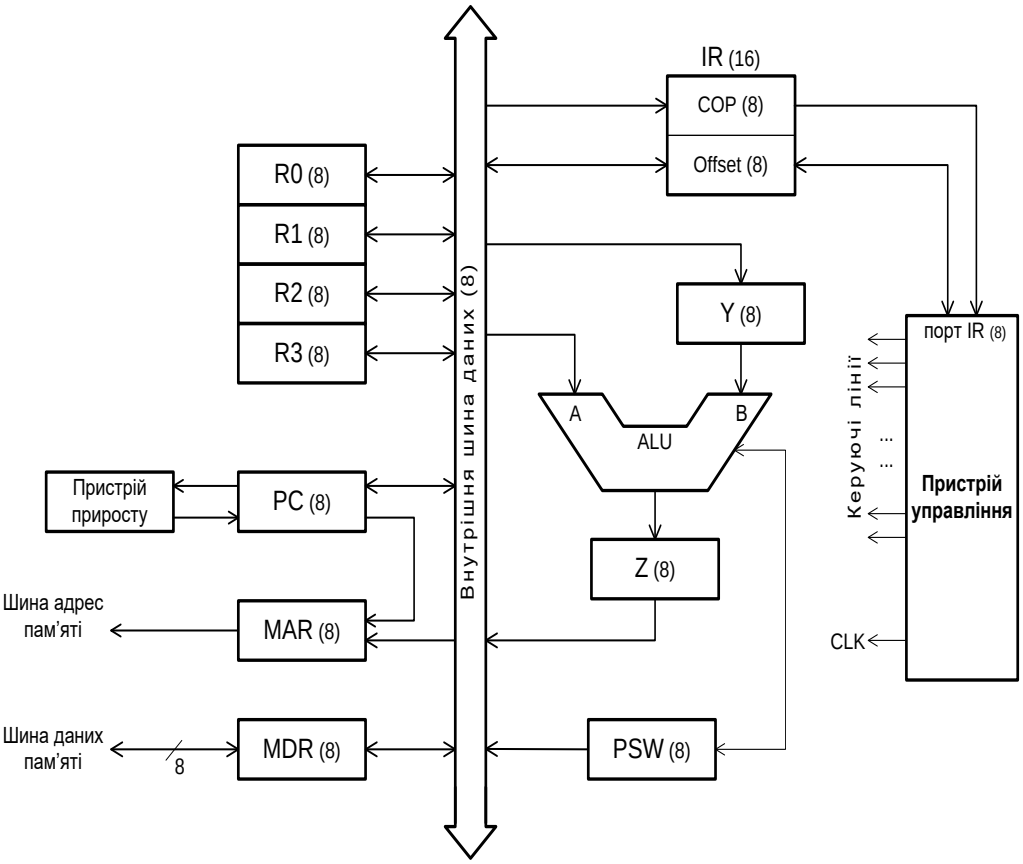
ДОДАТКИ

Додаток А

Загальна структура створеного нейропроцесорного елемента

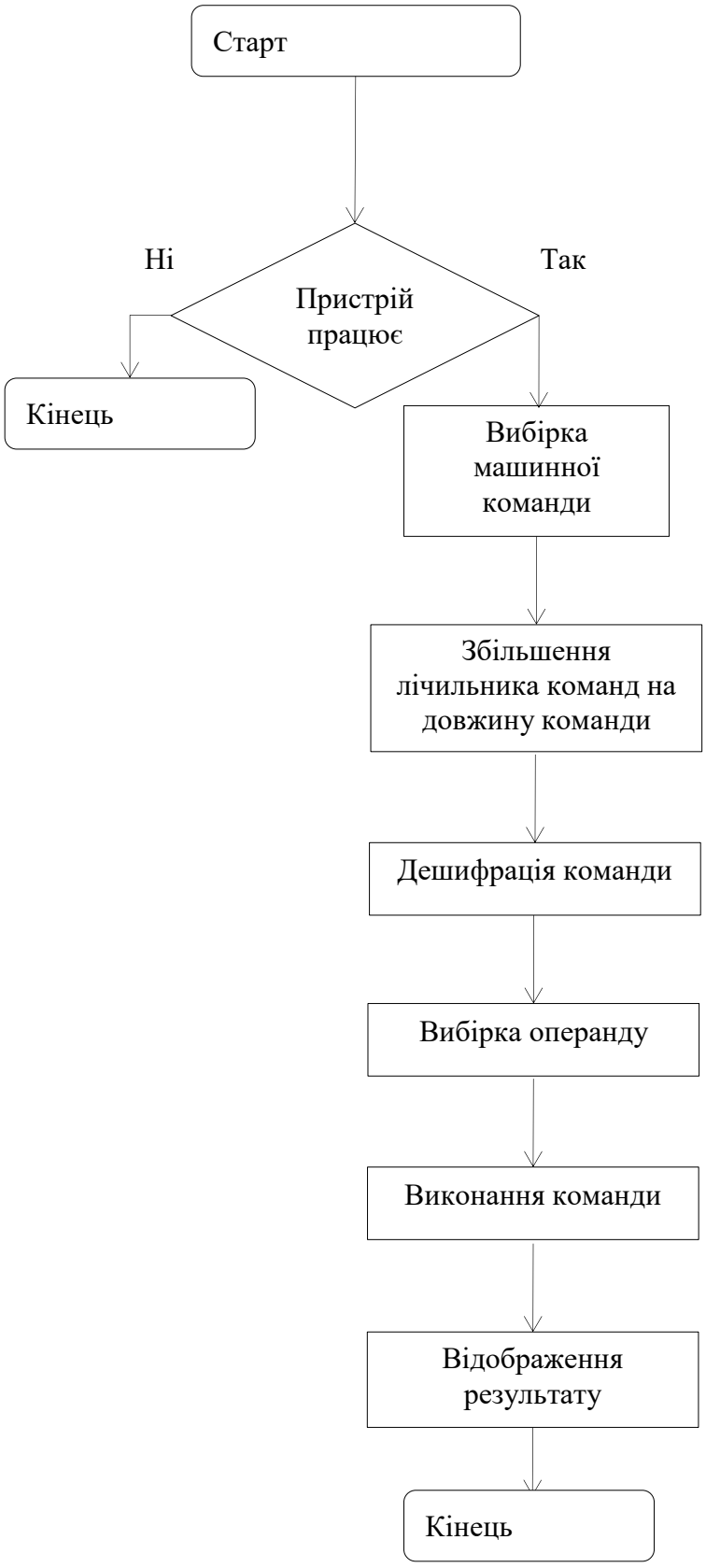


Розшифровка позначень на схемі

Символьне позначення	Опис
R0, R1, R2, R3	Регістри загального призначення, використовуються програмістами для потреб програм.
Z	Регістр, призначені для тимчасового збереження результату виконання певної операції арифметико-логічним пристроєм.
Y	Регістр для тимчасового збереження операнду, який потрібно обробити в АЛП.
IR	Instruction register – Регістр, в якому зберігається код машинної команди.
ALU	Arithmetical-logical unit - Арифметико-логічний пристрій, АЛП
PC	Program counter – Лічильник команд
MAR	Memory address register – Регістр адрес пам’яті
MDR	Memory data register – Регістр даних пам’яті
PSW	Program status word – Регістр стану

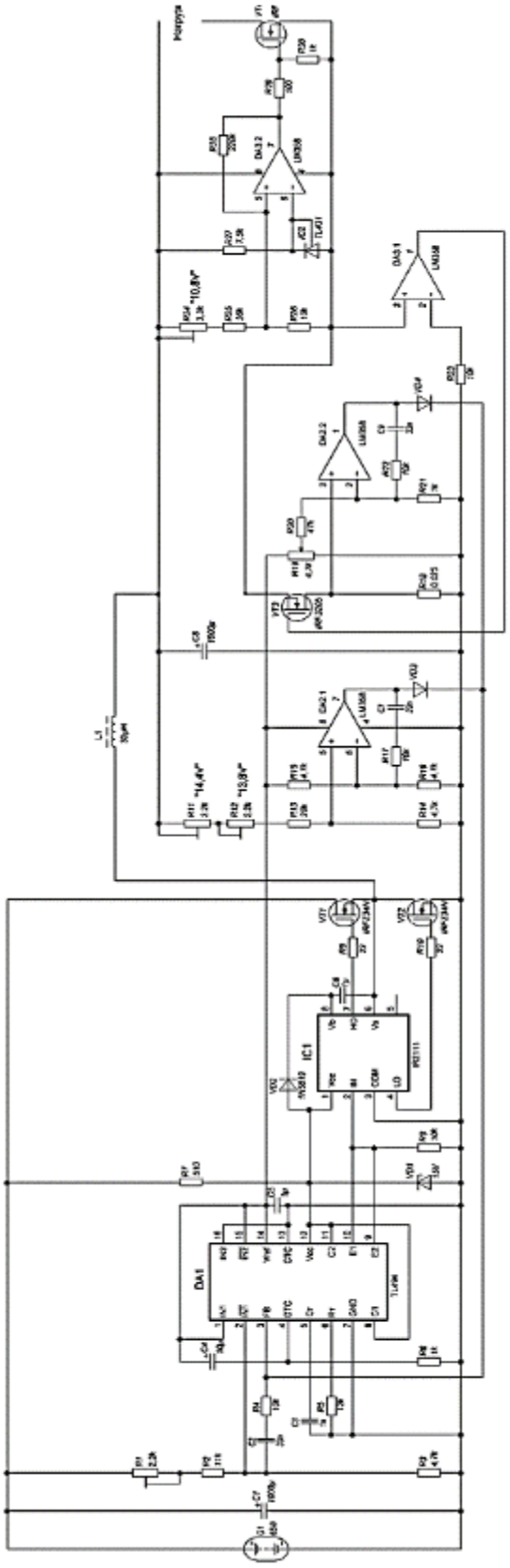
Додаток Б

Загальний алгоритм роботи логічного нейропроцесора



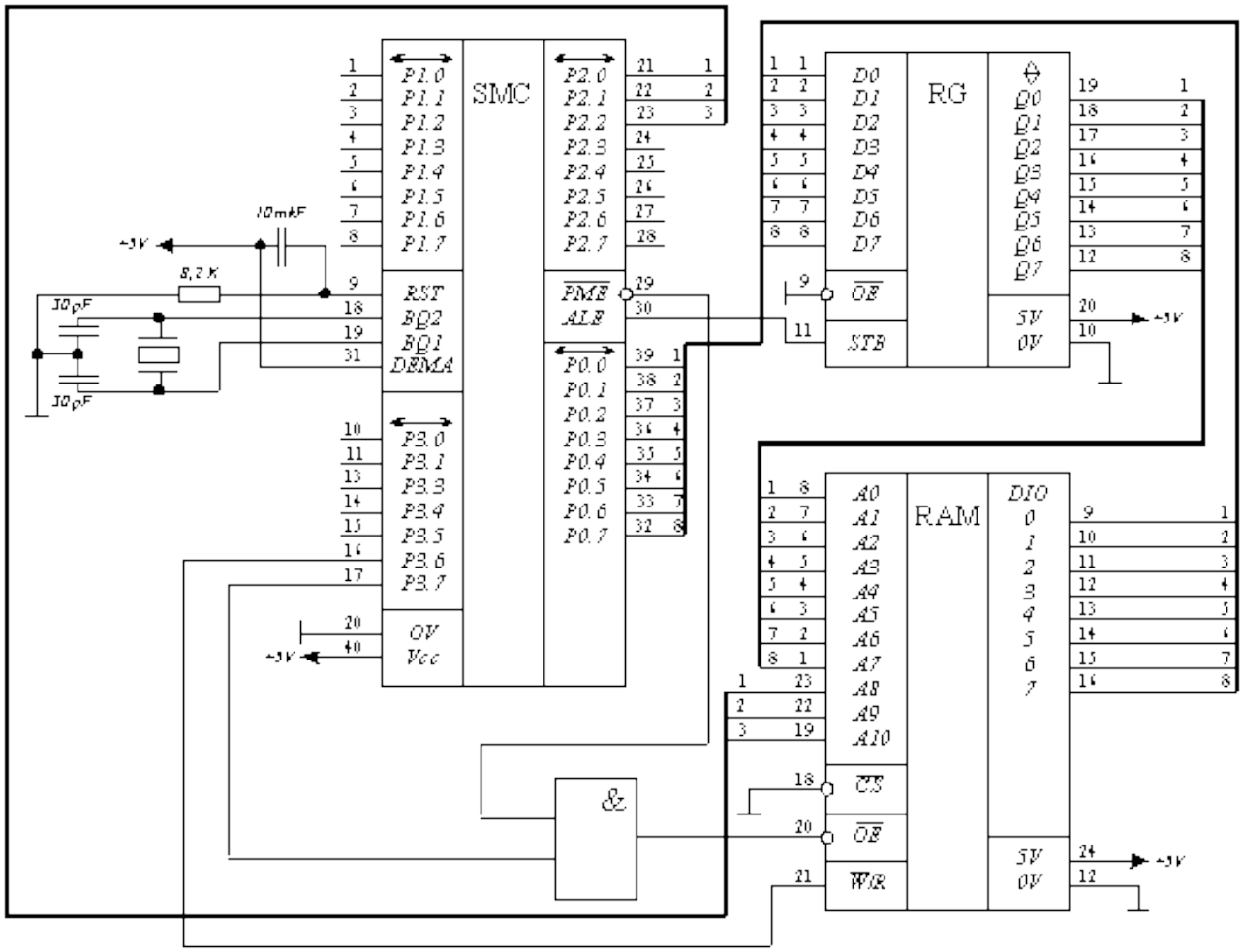
Додаток В

Схема електрична принципова імпульсного перетворювача для керування електроприводом



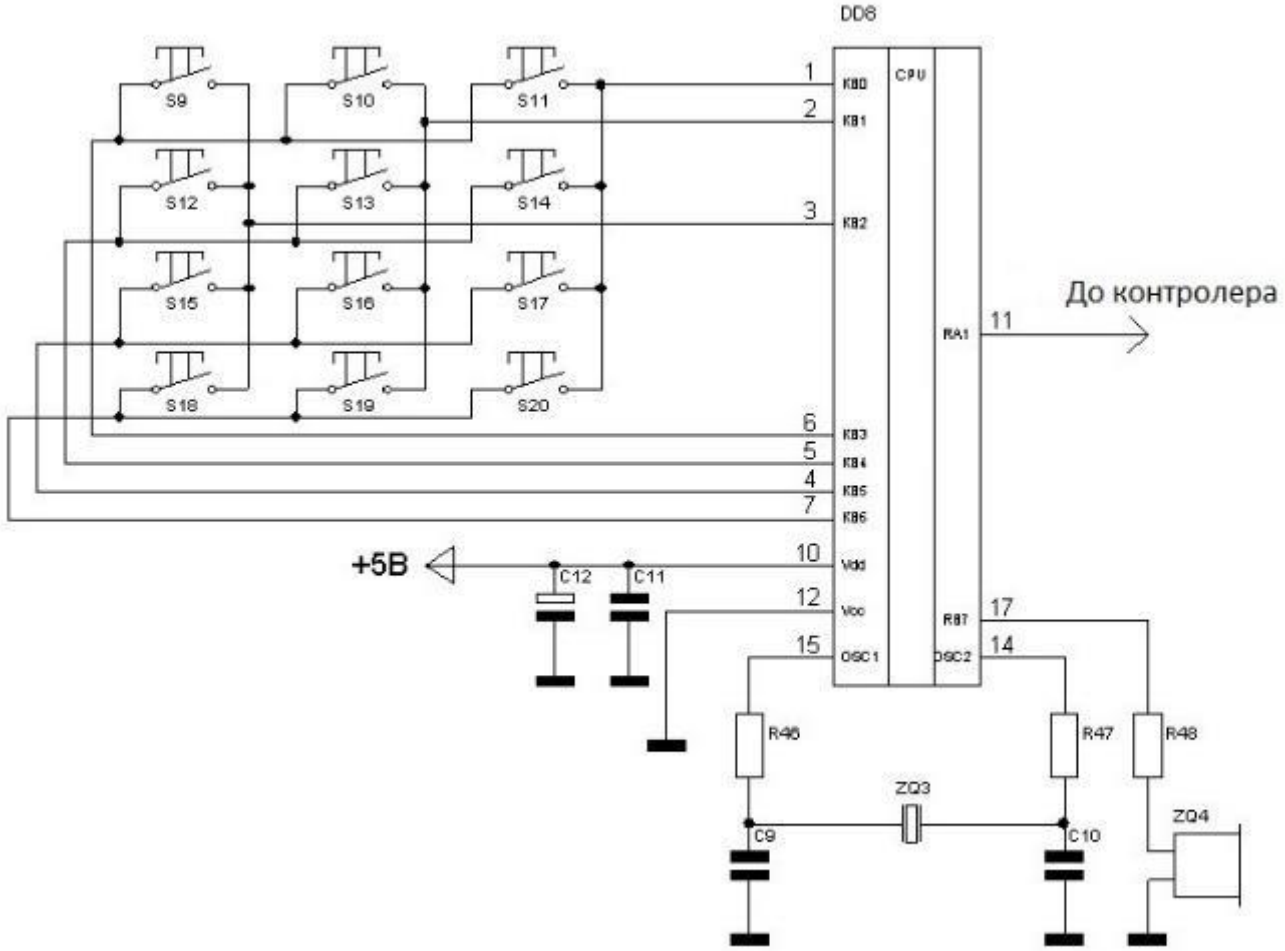
Додаток Г

Схема електрична принципова мікропроцесорного контролера



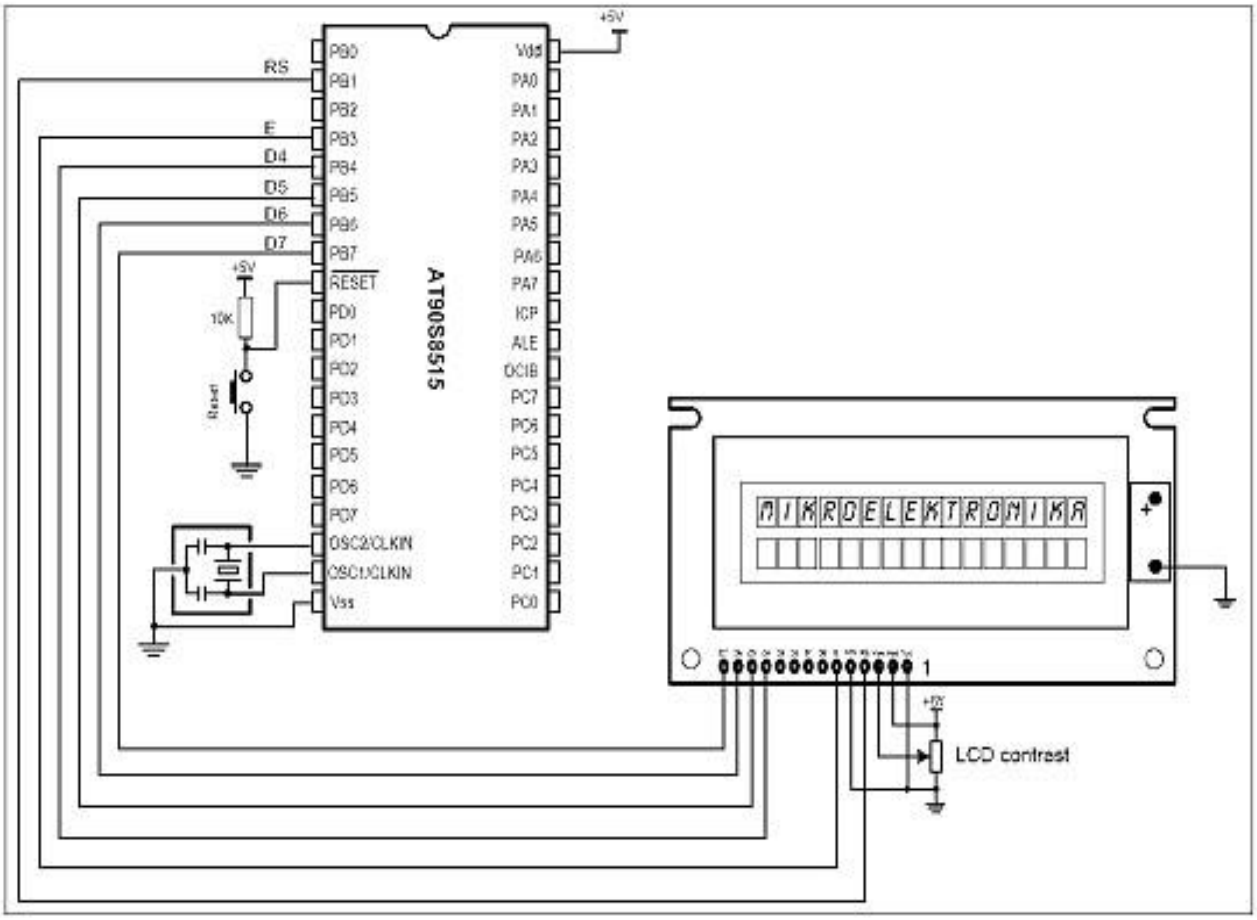
Додаток Д

Схема електрична принципова модуля клавіатури



Додаток Е

Схема електрична принципова LCD дисплея



Додаток Ж

Звіти синтезу проєкту логічного нейропроцесора для виконання логічних операцій на ПЛІС XC3S200–4-FT256 і XC7K160T – 3-FBG484

```
=========================================================
*                     Synthesis Options Summary
=========================================================
---- Source Parameters
Input File Name                    : "fir_filtr.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "fir_filtr"
Output Format                      : NGC
Target Device                      : xc3s200-4-ft256


=========================================================
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                                              : 2
 256x18-bit dual-port block RAM                     : 1
 256x20-bit single-port block RAM                   : 1
# Multipliers                                       : 1
 20x18-bit multiplier                               : 1
# Adders/Subtractors                                : 1
 8-bit subtractor                                   : 1
# Counters                                          : 1
 8-bit down counter                                 : 1
# Accumulators                                      : 1
 40-bit up accumulator                              : 1
# Registers                                         : 54
 Flip-Flops                                         : 54

=========================================================
=========================================================
Final Register Report

Macro Statistics
# Registers                                         : 108
 Flip-Flops                                         : 108

=========================================================
```

```
Device utilization summary:
---------------------------

Selected Device : 3s200ft256-4

 Number of Slices:                       79  out of   1920     4%
 Number of Slice Flip Flops:             89  out of   3840     2%
 Number of 4 input LUTs:                142  out of   3840     3%
 Number of IOs:                          40
 Number of bonded IOBs:                  40  out of    173    23%
    IOB Flip Flops:                      19
 Number of BRAMs:                         2  out of     12    16%
 Number of MULT18X18s:                    2  out of     12    16%
 Number of GCLKs:                         1  out of      8    12%
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 6.119ns (Maximum Frequency: 163.425MHz)
   Minimum input arrival time before clock: 5.061ns
   Maximum output required time after clock: 7.165ns
   Maximum combinational path delay: No path found
```

```
=========================================================
*                     Synthesis Options Summary
=========================================================
---- Source Parameters
Input File Name                    : "fir_filtr.prj"
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "fir_filtr"
Output Format                      : NGC
Target Device                      : xc7k160t-3-fbg484
```

```
=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                                                 : 3
 256x18-bit dual-port block RAM                        : 2
 256x20-bit single-port block Read Only RAM            : 1
# Multipliers                                          : 1
 20x18-bit multiplier                                  : 1
# Adders/Subtractors                                   : 2
 40-bit adder                                          : 1
 8-bit subtractor                                      : 1
# Registers                                            : 110
 Flip-Flops                                            : 110
# Multiplexers                                         : 60
 1-bit 2-to-1 multiplexer                              : 58
 18-bit 2-to-1 multiplexer                             : 1
 8-bit 2-to-1 multiplexer                              : 1

=========================================================================

=========================================================================
Final Register Report

Macro Statistics
# Registers                                            : 119
 Flip-Flops                                            : 119

=========================================================================

Device utilization summary:
---------------------------

Selected Device : 7k160tfbg484-3


Slice Logic Utilization:
 Number of Slice Registers:             118  out of  202800     0%
 Number of Slice LUTs:                  139  out of  101400     0%
    Number used as Logic:               139  out of  101400     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    193
   Number with an unused Flip Flop:      75  out of    193    38%
   Number with an unused LUT:            54  out of    193    27%
   Number of fully used LUT-FF pairs:    64  out of    193    33%
   Number of unique control sets:        58

IO Utilization:
 Number of IOs:                         40
 Number of bonded IOBs:                 40  out of    285    14%
    IOB Flip Flops/Latches:             19

Specific Feature Utilization:
 Number of Block RAM/FIFO:               1  out of    325     0%
    Number using Block RAM only:         1
 Number of BUFG/BUFGCTRLs:               1  out of     32     3%
 Number of DSP48E1s:                     1  out of    600     0%
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 5.489ns (Maximum Frequency: 182.166MHz)
   Minimum input arrival time before clock: 1.352ns
   Maximum output required time after clock: 0.511ns
   Maximum combinational path delay: No path found
```

Додаток З

Топ-файл VHDL моделі логічного нейропроцесора

```
library IEEE;
use IEEE. STD_LOGIC_1164. ALL;
use IEEE. STD_LOGIC_ARITH. ALL;
use IEEE. STD_LOGIC_UNSIGNED. ALL;
SPIM top module
One cycle MIPS TM.
Xilinx WebPack 9.1i
Submodules: IF_chip, ID_chip, EX_chip, MEM_chip, CTL_chip
entity A_SPIM is
port (clock: in std_logic;
reset: in std_logic;
PC: out std_logic_vector (7 downto 0));
end A_SPIM;
architecture spim_structure of A_SPIM is
component ID_chip
port (clock: in std_logic;
reset: in std_logic;
Instruction: in std_logic_vector (31 downto 0);
write_data: in std_logic_vector (7 downto 0);
RegWrite: in std_logic;
RegDst: in std_logic;
Immediate: out std_logic_vector (7 downto 0);
Rs: out std_logic_vector (7 downto 0);
Rt: out std_logic_vector (7 downto 0);
Opcode: out std_logic_vector (5 downto 0));
end component;
component IF_chip
port (clock: in std_logic;
reset: in std_logic;
Branch_Address: in std_logic_vector (7 downto 0);
PCsrc: in std_logic;
Instruction: out std_logic_vector (31 downto 0);
NPC: out std_logic_vector (7 downto 0);
PCview: out std_logic_vector (7 downto 0));
end component;
component EX_chip
port (Branch: in std_logic;
ALUSrc: in std_logic;
NPC: in std_logic_vector (7 downto 0);
Rs: in std_logic_vector (7 downto 0);
Rt: in std_logic_vector (7 downto 0);
Immediate: in std_logic_vector (7 downto 0);
ALUResult: out std_logic_vector (7 downto 0);
Branch_Address: out std_logic_vector (7 downto 0);
PCsrc: out std_logic);
end component;
component CTL_chip
port (Op: in std_logic_vector (5 downto 0);
RegDst: out std_logic;
ALUSrc: out std_logic;
MemtoReg: out std_logic;
```

```
RegWrite: out std_logic;
MemWrite: out std_logic;
Branch: out std_logic);
end component;
component MEM_chip
port (clock: in std_logic;
reset: in std_logic;
Memwrite: in std_logic;
MemtoReg: in std_logic;
address: in std_logic_vector (7 downto 0);
write_data: in std_logic_vector (7 downto 0);
writeback_data: out std_logic_vector (7 downto 0));
end component;
signal NPC_bus: std_logic_vector (7 downto 0);
signal Rs_bus: std_logic_vector (7 downto 0);
signal Rt_bus: std_logic_vector (7 downto 0);
signal OpCode_bus: std_logic_vector (5 downto 0);
signal Imm_bus: std_logic_vector (7 downto 0);
signal BrAddress_bus: std_logic_vector (7 downto 0);
signal ALUresult_bus: std_logic_vector (7 downto 0);
signal write_back_bus: std_logic_vector (7 downto 0);
signal Instruction_bus: std_logic_vector (31 downto 0);
signal Branch_wire: std_logic;
signal PCsrc_wire: std_logic;
signal RegWrite_wire: std_logic;
signal MemtoReg_wire: std_logic;
signal ALUSrc_wire: std_logic;
signal MemWrite_wire: std_logic;
signal RegDst_wire: std_logic;
begin
U_IF: IF_chip port map (
clock => clock,
reset => reset,
Branch_Address => BrAddress_bus,
PCsrc => PCsrc_wire,
Instruction => Instruction_bus,
NPC => NPC_bus,
PCview => PC);
U_ID: ID_chip port map (
clock => clock,
reset => reset,
instruction => Instruction_bus,
write_data => write_back_bus,
RegWrite => RegWrite_wire,
RegDst => RegDst_wire,
Rs => Rs_bus,
Rt => Rt_bus,
OpCode => OpCode_bus,
Immediate => Imm_bus);
U_EX: EX_chip port map (
Branch => Branch_wire,
ALUSrc => ALUSrc_wire,
NPC => NPC_bus,
Rs => Rs_bus,
```

```
Rt => Rt_bus,
Immediate => Imm_bus,
PCsrc => PCsrc_wire,
Branch_Address => BrAddress_bus,
ALUResult => ALUResult_bus);
U_CTL: CTL_chip port map (
Op => OpCode_bus,
RegDst => RegDst_wire,
ALUSrc => ALUSrc_wire,
MemtoReg => MemtoReg_wire,
RegWrite => RegWrite_wire,
MemWrite => MemWrite_wire,
Branch => Branch_wire);
U_MEM: MEM_chip port map (
clock => clock,
reset => reset,
MemWrite => MemWrite_wire,
MemtoReg => MemtoReg_wire,
writeback_data => write_back_bus,
address => ALUResult_bus,
write_data => Rt_bus);
end spim_structure;
```

Модуль керування (CTL)

```
-ctrl_chip
library IEEE;
use IEEE. STD_LOGIC_1164. all;
use IEEE. STD_LOGIC_ARITH. all;
entity ctrl_chip is
port (
Op: in STD_LOGIC_VECTOR (5 downto 0);
RegDst: out STD_LOGIC;
ALUSrc: out STD_LOGIC;
MemWrite: out STD_LOGIC;
MemtoReg: out STD_LOGIC;
RegWrite: out STD_LOGIC;
Branch: out STD_LOGIC
);
end ctrl_chip;
-}} End of automatically maintained section
architecture behav of ctl_chip is
signal R_format, lw, sw, beq: std_logic;
begin
R_format <= ( (not Op (5)) and (not Op (4)) and (not Op (3))
and
(not Op (2)) and (not Op (1)) and (not Op (0)));
lw <= (Op (5)) and (not Op (4)) and (not Op (3)) and
(not Op (2)) and (Op (1)) and (Op (0));
sw <= (Op (5)) and (not Op (4)) and (Op (3)) and
(not Op (2)) and (Op (1)) and (Op (0));
beq <= (not Op (5)) and (not Op (4)) and (not Op (3)) and
(Op (2)) and (not Op (1)) and (not Op (0));
RegDst <= R_format;
```

```
ALUSrc <= lw or sw;
MemtoReg <= lw;
RegWrite <= R_format or lw;
MemWrite <= sw;
Branch <= beq;
end behav;
```

Модуль виконання операцій (EXE)

```
EX-chip
library IEEE;
use IEEE. STD_LOGIC_1164. ALL;
use IEEE. STD_LOGIC_ARITH. ALL;
use IEEE. STD_LOGIC_UNSIGNED. ALL;
entity ex_chip is
port (Branch: in std_logic;
ALUSrc: in std_logic;
NPC: in std_logic_vector (7 downto 0);
Rs: in std_logic_vector (7 downto 0);
Rt: in std_logic_vector (7 downto 0);
Immediate: in std_logic_vector (7 downto 0);
ALUResult: out std_logic_vector (7 downto 0);
Branch_Address: out std_logic_vector (7 downto 0);
PCsrc: out std_logic);
end ex_chip;
architecture behav of ex_chip is
signal Branch_Address_tmp: std_logic_vector (8 downto 0);
signal Zero: std_logic;
begin
ALUresult <= (Rs + Rt) when (ALUsrc = '0') else (Rs +
Immediate);
Branch_Address_tmp <= ('0' & NPC) + (Immediate (6 downto 0)
&'0'&'0');
Branch_Address <= Branch_Address_tmp (7 downto 0);
Zero <= '1' when (Rs = Rt) else '0';
PCsrc <= Branch and Zero;
end behav;
```

Модуль декодування інструкцій (ID)

```
ID chip
library IEEE;
use IEEE. STD_LOGIC_1164. ALL;
use IEEE. STD_LOGIC_ARITH. ALL;
use IEEE. STD_LOGIC_UNSIGNED. ALL;
entity ID_chip is
port (clock: in std_logic;
reset: in std_logic;
RegWrite: in std_logic;
RegDst: in std_logic;
instruction: in std_logic_vector (31 downto 0);
write_data: in std_logic_vector (7 downto 0);
Immediate: out std_logic_vector (7 downto 0);
```

```
     Rs: out std_logic_vector (7 downto 0);
     Rt: out std_logic_vector (7 downto 0);
     OpCode: out std_logic_vector (5 downto 0));
     end ID_chip;
     architecture behav of ID_chip is
     signal   reg1,   reg2,   reg3,   reg4,   reg5,   reg6,   reg7:
std_logic_vector (7 downto 0);
     signal reg1tmp, reg2tmp, reg3tmp, reg4tmp,
     reg5tmp, reg6tmp, reg7tmp: std_logic_vector (7 downto 0);
     signal reg1wr, reg2wr, reg3wr, reg4wr,
     reg5wr, reg6wr, reg7wr: std_logic;
     signal rd_addr1, rd_addr2: std_logic_vector (4 downto 0);
     signal wr_addr_ALUop, wr_addr_LWop,
     wr_addr: std_logic_vector (4 downto 0);
     begin
     OpCode <= Instruction (31 downto 26);
     rd_addr1 <= Instruction (25 downto 21);
     rd_addr2 <= Instruction (20 downto 16);
     wr_addr_ALUop <= Instruction (15 downto 11);
     wr_addr_LWop <= Instruction (20 downto 16);
     Immediate <= Instruction (7 downto 0);
     with rd_addr1 (4 downto 0) select
     Rs <= x «00» when «00000»,
     reg1 when «00001»,
     reg2 when «00010»,
     reg3 when «00011»,
     reg4 when «00100»,
     reg5 when «00101»,
     reg6 when «00110»,
     reg7 when «00111»,
     x'FF» when others;
     with rd_addr2 (4 downto 0) select
     Rt <= x «00» when «00000»,
     reg1 when «00001»,
     reg2 when «00010»,
     reg3 when «00011»,
     reg4 when «00100»,
     reg5 when «00101»,
     reg6 when «00110»,
     reg7 when «00111»,
     x'FF» when others;
     wr_addr <= wr_addr_ALUop when RegDst='1' else wr_addr_LWop;
     reg1wr<='1' when ( (wr_addr=«00001») and (RegWrite='1')) else
'0';
     reg2wr<='1' when ( (wr_addr=«00010») and (RegWrite='1')) else
'0';
     reg3wr<='1' when ( (wr_addr=«00011») and (RegWrite='1')) else
'0';
     reg4wr<='1' when ( (wr_addr=«00100») and (RegWrite='1')) else
'0';
     reg5wr<='1' when ( (wr_addr=«00101») and (RegWrite='1')) else
'0';
     reg6wr<='1' when ( (wr_addr=«00110») and (RegWrite='1')) else
'0';
```

```
    reg7wr<='1' when ( (wr_addr=«00111») and (RegWrite='1')) else
'0';
    reg1tmp <= write_data when reg1wr='1' else reg1;
    reg2tmp <= write_data when reg2wr='1' else reg2;
    reg3tmp <= write_data when reg3wr='1' else reg3;
    reg4tmp <= write_data when reg4wr='1' else reg4;
    reg5tmp <= write_data when reg5wr='1' else reg5;
    reg6tmp <= write_data when reg6wr='1' else reg6;
    reg7tmp <= write_data when reg7wr='1' else reg7;
    process
    begin
    wait until clock'event and clock='1';
    if reset='1' then
    reg1 <= x'A1»;
    reg2 <= x'A2»;
    reg3 <= x'A3»;
    reg4 <= x'A4»;
    reg5 <= x'A5»;
    reg6 <= x'A6»;
    reg7 <= x'A7»;
    else
    reg1 <= reg1tmp;
    reg2 <= reg2tmp;
    reg3 <= reg3tmp;
    reg4 <= reg4tmp;
    reg5 <= reg5tmp;
    reg6 <= reg6tmp;
    reg7 <= reg7tmp;
    end if;
    end process;
    end behav;
```

Модуль IF

```
    IF chip
    library IEEE;
    use IEEE. STD_LOGIC_1164. ALL;
    use IEEE. STD_LOGIC_ARITH. ALL;
    use IEEE. STD_LOGIC_UNSIGNED. ALL;
    entity IF_chip is
    port (clock: in std_logic;
    reset: in std_logic;
    PCsrc: in std_logic;
    Branch_Address: in std_logic_vector (7 downto 0);
    PCview: out std_logic_vector (7 downto 0);
    Instruction: out std_logic_vector (31 downto 0);
    NPC: out std_logic_vector (7 downto 0));
    end IF_chip;
    architecture behav of IF_chip is
    signal PC: std_logic_vector (7 downto 0);
    signal Input_PC: std_logic_vector (7 downto 0);
    signal NPC_temp: std_logic_vector (7 downto 0);
    Test Program
    constant rom0: std_logic_vector (31 downto 0): =x «8c040000»;
```

```
     lw $4, 0 ($0)
     constant   rom1:   std_logic_vector   (31   downto   0):   =   x
«8c050001»;
     lw $5, 1 ($0)
     constant   rom2:   std_logic_vector   (31   downto   0):   =   x
«00852020»;
     add $4, $4, $5
     constant rom3: std_logic_vector (31 downto 0): = x’ac040000»;
     sw $4, 0 ($0)
     constant   rom4:   std_logic_vector   (31   downto   0):   =   x
«1080fffb»;
     beq $4, $0, - 20
     constant   rom5:   std_logic_vector   (31   downto   0):   =   x
«1084ffff»;
     beq $4, $4, - 4
     constant   rom6:   std_logic_vector   (31   downto   0):   =   x
«00000025»;
     or $0, $0, $0
     constant rom7: std_logic_vector (31 downto 0): =x «00000025»;
     or $0, $0, $0
     begin
     PCview <= PC;
     NPC_temp (7 downto 2) <= PC (7 downto 2) + 1;
     NPC_temp (1 downto 0) <= b «00»;
     NPC <= NPC_temp;
     Input_PC <= Branch_Address when PCsrc = '1' else NPC_temp;
     process
     begin
     wait until (clock’event) and (clock='1');
     If reset='1' then
     PC <= x «00»;
     else
     PC <= Input_PC;
     end if;
     end process;
     process (PC)
     begin
     case PC (7 downto 2) is
     when «000000» => instruction <= rom0;
     when «000001» => instruction <= rom1;
     when «000010» => instruction <= rom2;
     when «000011» => instruction <= rom3;
     when «000100» => instruction <= rom4;
     when «000101» => instruction <= rom5;
     when «000110» => instruction <= rom6;
     when «000111» => instruction <= rom7;
     when others => instruction <= x «00000000»;
     end case;
     end process;
     end behav;
```

Модуль пам’яті (MEM)

```
     MEM-chip
```

```
library IEEE;
use IEEE. STD_LOGIC_1164. ALL;
use IEEE. STD_LOGIC_ARITH. ALL;
use IEEE. STD_LOGIC_UNSIGNED. ALL;
entity MEM_chip is
port (clock: in std_logic;
reset: in std_logic;
MemWrite: in std_logic;
MemToReg: in std_logic;
address: in std_logic_vector (7 downto 0);
write_data: in std_logic_vector (7 downto 0);
writeback_data: out std_logic_vector (7 downto 0));
end MEM_chip;
architecture behav of MEM_chip is
signal read_data: std_logic_vector (7 downto 0);
signal cell0, cell1: std_logic_vector (7 downto 0);
signal tmp0, tmp1: std_logic_vector (7 downto 0);
signal wr0, wr1: std_logic;
begin
read_data <= cell0 when address=x «00» else
cell1 when address=x «01» else
x’FF»;
writeback_data <= address when MemtoReg='0' else read_data;
wr0 <= '1' when MemWrite = '1' and address (0) = '0' else
'0';
wr1 <= '1' when MemWrite = '1' and address (0) = '1' else
'0';
tmp0 <= write_data when wr0 ='1' else cell0;
tmp1 <= write_data when wr1 ='1' else cell1;
process
begin
wait until clock’event and clock='1';
if (reset = '1') then
cell0 <= x «02»;
cell1 <= x’FE»;
else
cell0 <= tmp0;
cell1 <= tmp1;
end if;
end process;
end behav;
```

Автоматично згенерований файл (Test Bench)

```
LIBRARY ieee;
USE ieee. std_logic_1164. ALL;
USE ieee. std_logic_unsigned. all;
USE ieee. numeric_std. ALL;
ENTITY tb_vhd IS
END tb_vhd;
ARCHITECTURE behavior OF tb_vhd IS
Component Declaration for the Unit Under Test (UUT)
COMPONENT A_SPIM
PORT (
```

```
clock: IN std_logic;
reset: IN std_logic;
PC: OUT std_logic_vector (7 downto 0)
);
END COMPONENT;
-Inputs
SIGNAL clock: std_logic: = '0';
SIGNAL reset: std_logic: = '1';
-Outputs
SIGNAL PC: std_logic_vector (7 downto 0);
BEGIN
Instantiate the Unit Under Test (UUT)
uut: A_SPIM PORT MAP (
clock => clock,
reset => reset,
PC => PC
);
clock <= not clock after 50ns;
reset <= '0' after 180ns;
tb: PROCESS
BEGIN
Wait 100 ns for global reset to finish
wait for 100 ns;
Place stimulus here
wait; - will wait forever
END PROCESS;
END;
```

Додаток И

Програма керування мікропроцесорного контролера

```
LD% I0.0.0 ANDN% M4 S% M0 BLK% TM0 LD% M0 IN OUT_BLK LD Q S% M1 END_BLK BLK% TM1 LD% M1 IN OUT_BLK LD Q R% M0 R% M1 END_BLK LD% Q0.0.1 OR% Q0.0.2 OR% Q0.0.3 OR% Q0.0.4 S% M4 LD% I0.0.1 R% M4 LD% M4 S% M5 BLK% TM2 LD% M5 IN OUT_BLK LD Q S% M² END_BLK BLK% TM1 LD% M² IN OUT_BLK
    LD Q R% M5 R% M² END_BLK LD% M1 OR% M² ST% M³ LD% I0.0.0 [%MW1:=%MW100] LD% I0.0.0 [%MW2:=%MW101] LD% I0.0.0 [%MW3:=%MW102] LD% I0.0.0 [%MW4:=%MW103] LD% I0.0.0 [%MW5:=%MW1] LD% I0.0.0 [%MW6:=%MW1-%MW3] LD% I0.0.0 [%MW7:=%MW2+%MW4] LD% I0.0.0 [%MW8:=%MW2-%MW4] LD [%IW0.1.0> %MW5] ST% M6 LD [%IW0.1.0< %MW6] ST% M7 LD% I0.0.0 [%MW0:=%IW0.1.2 / 10] LD% I0.0.0 [%MW12:=%IW0.1.0+ 0.2% IW0.1.3] [%MW9:=%IW0.1.1 -%MW0] LD [%MW9>%MW7] ST% M8 LD [%MW9<%MW8] ST% M9 LD% I0.0.0 [PID 0] LD% I0.0.0 [PID 1] LD% I0.0.0 [%MW12:=%MW10*% KW1] LD% I0.0.0 [%MW13:=%MW11*% KW1] LD [%MW12>=0] AND% M6 ST% M12 BLK% TM4 LD% M6 IN OUT_BLK LD Q ST% M10 END_BLK LD% M10 AND% M³ ST% M11 LD% M12 ANDN% M11 S% M13 LD% M7 R% M13 LD% M13 AND% M6 AND% M11 ANDN% I0.0.2
    ST% Q0.0.1 LD [%MW12<=0] AND% M7 ST% M16 BLK% TM5 LD% M7 IN OUT_BLK LD Q ST% M14 END_BLK LD% M14 AND% M³ ST% M15 LD% M16 ANDN% M15 S% M17 LD% M6 R% M17 LD% M17 AND% M7 AND% M15 ANDN% I0.0.2 ST% Q0.0.2 BLK% TM6 LD% M8 IN OUT_BLK LD Q ST% M18 END_BLK LD% M18 AND% M³ ST% M19 LD [%MW13>=0] AND% M8 ST% M20 LD% M20
    ANDN% M19 S% M21 LD% M9 R% M21 LD% M21 AND% M8 AND% M19 ANDN% I0.0.2 ST% Q0.0.3 BLK% TM7 LD% M9 IN OUT_BLK LD Q ST% M22 END_BLK LD% M22 AND% M³ ST% M23 LD [%MW13<=0] AND% M9 ST% M24 LD% M24 ANDN% M23 S% M25 LD% M8 R% M25 LD% M25 AND% M23 AND% M9 ANDN% I0.0.2 ST% Q0.0.4 LD% I0.0.2 ST% Q0.0.5 LD% M³ ST% Q0.0.6 END
```

Додаток К

Лістинг програми мікроконтролера PIC16F84–041 / p

```
     ;*********************************************************
;********* ОПРЕДЕЛЕНИЕ ПЕРЕМЕННЫХ ************************
     w_temp EQU 0x0C; variable used for context saving
     status_temp EQU 0x0D; variable used for context saving
          cblock 0x0E
     cnt0, cnt1, cnt2         ;    счетчики    для    программируемой
задержки
     key_tmp, scan            ; для хранения кода нажатой клавиши
     cnt_body0, cnt_body1          ; счетчики числа проходов
     cnt_adc             ; счетчик измерения АЦП
     flags                    ; внутренние флаги (см ниже)
     cod0, cod1, cod2, cod3        ;
     tmp_snd, tmp_snd1        ; хранение признака тона
     cnt10                    ;

          endc

     razr      equ 0          ; флаг flags
     sost      equ 1          ; состояние (1-закрыт)

     ;
*************************************************************
     ;************ РАСПРЕДЕЛЕНИЕ ПАМЯТИ ЕЕ *********************
     addr_cod  equ 0          ;
     addr_razr equ 0x10  ;
     ; *******************************************************
     ;*******РАСПРЕДЕЛЕНИЕ ПОРТОВ ****************************
     psnd      equ PORTA ; выход звукового сигнала
     snd       equ 4
     prele          equ PORTA ;
     rele0          equ 0
     rele1          equ 1
     padc      equ PORTA ; псевдоАЦП
     adc       equ 2
     tris_adc  equ TRISA
     pkey      equ PORTB ; порт клавиатуры
     clop      equ 7          ; кнопка на внутренней стороне ОТКР /
ЗАКР
     line0          equ 0          ; линия сканирования 0
     line1          equ 1          ; линия сканирования 1
     line2          equ 2          ; линия сканирования 2
     line3          equ 3          ; линия сканирования 3
     otkl0          equ 4          ; линия отклика 0
     otkl1          equ 5          ; линия отклика 1
     otkl2          equ 6          ; линия отклика 2

     ;*********************************************************
     ********** ПОДПРОГРАММА ОБРАБОТКИ ПРЕРЫВАНИЙ ************
     ;*********************************************************
               ORG 0x000; processor reset vector
               goto main; go to beginning of program
```

```
                ORG 0x004; interrupt vector location
                movwf w_temp; save off current W register contents
                movf STATUS, w; move status register into W register
                movwf     status_temp; save off contents of STATUS
register
     ; isr code can go here or be located as a call subroutine
elsewhere

                movf  status_temp,  w;  retrieve  copy  of  STATUS
register
                movwf     STATUS; restore pre-isr STATUS register
contents
                swapf w_temp, f
                swapf w_temp, w; restore pre-isr W register contents
                retfie; return from interrupt

     ;*****************************************************
     *****************МЕСТО РАСПОЛОЖЕНИЯ ПОДПРОГРАММ **********
     ;*****************************************************
     ; П / П ПРОГРАММИРУЕМОЙ ЗАДЕРЖКИ --------------------
     ; задержка в милисекундах во wreg
     ; Входные: WREG – задержка в десяткал милисекунд
     ; 1 во WREG соответствует 10 мс
     delay
                movwf cnt0
     delay1          movlw. 12
                movwf cnt1
     delay3          movlw. 207
                movwf cnt2
     delay2          clrwdt
                decfsz cnt2, f
                goto delay2
                decfsz cnt1, f
                goto delay3
                decfsz cnt0, f
                goto delay1
                return

     ; ЗВУКОВОЙ ПИСК ----------------
     ; сигнал хитрой формы длительностью порядка 0,15 секунды
     ; WREG – параметр изменения тона. Должен быть в пределах:
1......10
     ; 1000 Гц => 500 мкс + 500 мкс
     ; За 0.15 секунды пройдет 150 периодов
     pisk
                movwf tmp_snd       ; сохраним
                bcf STATUS, C
                rlf tmp_snd, f      ; умножим на 2
                movf tmp_snd, w
                movwf tmp_snd1
                movlw. 100
                movwf cnt2          ;  для  определения  длительности
импульса
```

```
                movlw. 100          ;     определение    длительности
сигнала
                movwf cnt0          ; ПЕРВАЯ треть сигнала
     pisk2
                snd_on
                call pisks
                snd_off
                call pisks
                movf tmp_snd, f          ; проверка на 0
                btfsc STATUS, Z
                goto pisk3
                decf tmp_snd, f
                incf cnt2, f
     pisk3           decfsz cnt0, f
                goto pisk2

                movlw. 50
                movwf cnt0
     pisk4
                snd_on
                call pisks
                snd_off
                call pisks
                movf tmp_snd1, f         ; проверка на 0
                btfsc STATUS, Z
                goto pisk5
                decf tmp_snd1
                decf cnt2, f
     pisk5           decfsz cnt0, f
                goto pisk4
                return

     pisks                       ; п / п задержки
                movf cnt2, w
                movwf cnt1
     pisks1          clrwdt
                nop
                decfsz cnt1, f
                goto pisks1
                return

     ;*************************************************************
     ;*********** НАЧАЛО ОСНОВНОГО КОДА ПРОГРАММЫ ************
     ;*************************************************************
     main
     ; инициализация порта A
     ; 1 - вход
     PORTA_NAPR     EQU B'00001000'
     PORTB_NAPR     EQU B'11110000'
               BANKSEL PORTA
               CLRF PORTA
               BANKSEL TRISA
               MOVLW PORTA_NAPR
               MOVWF TRISA
```

```
                BANKSEL PORTA
                CLRF PORTA

        ; инициализация порта B
                CLRF PORTB
                BANKSEL TRISB
                MOVLW PORTB_NAPR
                MOVWF TRISB
                BANKSEL PORTB
                CLRF PORTB
        ; общие
                CLRWDT
                BANKSEL OPTION_REG
                MOVLW B'01011111'   ; предделитель перед WDT с коэф
1:128
                MOVWF OPTION_REG    ;     подтягивающие     резисторы
включены
                BANKSEL TMR0
                clrf TMR0

        ; начальные установки
                clrf cnt_body0
                clrf cnt_body1
                clrf flags
                snd_off
                rele_off
                adc_off
                clrf EEADR
                movlw 0xFF
                movwf pkey

        ; начальный писк
                movlw. 10
                call pisk
                movlw. 25
                call delay
                movlw. 5
                call pisk
                movlw. 25
                call delay
                movlw 1
                call pisk
                movlw 1
                call pisk
                movlw 1
                call pisk

        ;*************************************************************
        ;*****************ТЕЛО РАБОЧЕГО ЦИКЛА *********************
        ;*************************************************************
        body
                banksel cnt_body0   ; счетчик количества проходов
                incfsz cnt_body0, f ; счетчик двухбайтный
                goto body1
```

```
                incf cnt_body1, f
     body1           btfss cnt_body1,5   ;  проверка  на  достижение
заданного
                goto body3          ; интервала времени (bit3)
                clrf cnt_body0
                clrf cnt_body1
                btfss flags, razr   ;
                goto body2          ;
                movlw. 3       ; звуковой сигнал
                call pisk
                movlw. 25
                call delay
                movlw. 3
                call pisk
                goto body3
     body2
                clrf pkey      ; подготовка ко сну
                bcf INTCON, RBIF
                bcf INTCON, GIE
                bsf INTCON, RBIF
                sleep
                nop
     body3                      ;
                btfsc flags, sost   ;  проверка  действия  при  нажати
кнопки
                goto body4

                call keyboard       ;
                movwf key_tmp       ;
                incfsz key_tmp, f   ;   проверка   нажатия   основной
клавиатуры
                goto body5
                btfsc pkey, clop        ;      проверка      нажатия
внутренней кнопки
                goto body10         ; ничто не нажато
     body5           call close          ; процедура закрывания
                bsf flags, sost
                movlw. 30
                call delay
                goto body10

     body4                      ;
                btfss pkey, clop        ;    проверка    внутренней
кнопки
                goto body18

                movlw. 40
                movwf cnt10         ; счетчик сремени опроса
     body6           call keyboard       ; первая цифра кода
                addlw 1
                btfss STATUS, Z
                goto body7
                decfsz cnt10, f
                goto body6
```

```
body8          movlw 1
          call pisk      ; отказ от дальнейшего ввода
          movlw. 15
          call delay
          movlw 6
          call pisk      ; отказ от дальнейшего ввода
          movlw. 15
          call delay
          movlw 1
          call pisk      ; отказ от дальнейшего ввода
          movlw. 15
          call delay
          goto body10
body7          movf key_tmp, w
          movwf cod0
          movlw. 40
          movwf cnt10
body9          call keyboard       ; вторая цифра
          addlw 1
          btfss STATUS, Z
          goto body11
          decfsz cnt10, f
          goto body9
          goto body8
body11         movf key_tmp, w
          movwf cod1
          movlw. 40
          movwf cnt10
body13         call keyboard       ; третья цифра
          addlw 1
          btfss STATUS, Z
          goto body12
          decfsz cnt10, f
          goto body13
          goto body8
body12         movf key_tmp, w
          movwf cod2
          movlw. 40
          movwf cnt10
body15         call keyboard       ; четвертая цифра
          addlw 1
          btfss STATUS, Z
          goto body14
          decfsz cnt10, f
          goto body15
          goto body8
body14         movf key_tmp, w
          movwf cod3

          movlw. 4       ; проверка на код в ЕЕ
          movwf cnt10
          movlw cod0
          movwf FSR
          movlw addr_cod
```

```
                movwf EEADR
     body16
                call read_ee
                movf INDF, w
                subwf EEDATA, w
                btfss STATUS, Z
                goto body17         ; переход к проверке на другой
код.
                incf FSR, f
                incf EEADR, f
                decfsz cnt10, f
                goto body16

     body18
                movlw. 5
                call pisk
                movlw. 5
                call pisk
                movlw. 5
                call pisk
                call open; процедура ОТКРЫВАНИЯ
                bcf flags, sost
                movlw. 30
                call delay
                goto body10
```