VEMA
Industrial electronic systems
Bulgaria, Pleven 5800, 27 Osogovo str.
tel./fax: +359/64/870172, tel.: +359/64/870170
e-mail: office@vema-bg.com http://www.vema-bg.com

PROGRAMMABLE LOGIC CONTROLLERS

VPC106
VPC108A4
VPC1410
VPC1812
VPC2416
VPC3224L
VPC4030
VPC3224A63
VPC4030A84

USER'S MANUAL

I. Introduction

The *VPC* programmable logic controllers (PLC) are galvanic isolated from the outside electric environment by opto-isolated discrete inputs 24V/10mA, and opto-isolated outputs 24V/2A.

All *VPC* controllers have identical structure to ease the user.

The number of the inputs and the outputs of each *VPC* PLC are shown on the table in the next chapter. Three of the controllers have additional analog inputs and outputs: ***VPC3224A63*** is equipped with 6 analog inputs 0-20mA/0-10V and 3 analog outputs 0-20mA/0-10V. ***VPC108A4*** has 4 analog inputs 0-20mA/0-10V. ***VPC4030A84*** PLC can connect to external ***VPCIO*** module to expand the number of digital inputs and outputs, or add up to 8 analog inputs and 4 analog outputs.

All *VPC* PLCs have non-volatile EEPROM memory to store discrete and decimal registers, and FLASH memory for the user program.

VPC PLCs are equipped with inbuilt programming console which enables the user of setting technological parameters in running mode, and editing the controller's program. The *VPC* programming language is easy to use mnemonics, allowing for quick creation of end-user programs.

Using the RS232C interface, the *VPC* PLCs can communicate with PCs. The computer needs to have **VPC Host Interface** application to give the user possibility to create, edit and transfer *VPC* programs to and from the *VPC* PLC. The programs with **VPC Host Interface** can be created both in mnemonic and ladder diagrams.

II. Technical specifications

PLC name	VPC106	VPC108A4	VPC1410	VPC1812	VPC2416	VPC3224L	VPC3224A63	VPC4030	VPC4030A84
discrete inputs	10	10	14	18	24	32	32	40	40
discrete outputs	6	8	10	12	16	24	24	30	30
analog inputs		4					6		8*
analog outputs							3		4*

* VPC84IO expansion module is required

-Discrete inputs:
- Input voltage - 18 to 30 VDC
- Input current - 10 mA at 24V

-Discrete outputs:
- Maximal output current - 2A
- Maximal output voltage - 30V
- Maximal voltage of a set output - 1.4V at 2A

-Analog inputs - 0-20mA/0-10V
- Analog outputs - 0-20mA/0-10V
- Timers (00.00 to 99.99 s) - 32 (see.*Remark*)
- Counters (0000 to 9999) - 32 (see.*Remark*)
- Auxiliary discrete registers (ON/OFF) -96
- Auxiliary decimal parameters (0000-9999) -78
- Number of program lines - 2048
- Maximal duration of a program cycle (for 2048 lines) - 7 ms
- Programming console - inbuilt
- Indication:
  - LCD display, 4 lines, 16 characters (**VPC108A4** and **VPC3224A63** only)
    2 lines, 16 characters (all other *VPC* models)
  - LEDs to show which inputs/outputs are driven ON (**VPC2416** and higher)
- Communication interface to PC - RS232 C

- Modes:
  - working mode with monitoring - **RUN & MONITOR**
  - edit and search - **PROGRAM**
- Power supply - 100 to 250 VAC
- Ambient temperature (working) - 0 to 50 °C
- Ambient temperature (store) - (-10) to 60 °C

III. Registers and operands

- discrete inputs - *i00,...,i39*;
- discrete outputs - *o00,...,o29*;
- timers with time value 00.00 to 99.99 s - *t00,...,t31*;
- counters (0000- 9999) - *c00,...,c31*;

Remark. Timers and counters must have different indexes, because they share mutual memory bank. The total number of timers and counters is 32. Each of the numbers 0,...,31 is assign either to a timer, or to a counter. See next chapter how to toggle assignment between timer and counter.

- auxiliary registers (discrete):
  - readable and writable - *r00,...,r87*;
  - read-only (system):
    - *r88(RUN)* - always ON, if it is lowered to OFF in PROGRAM mode, the *VPC* controller will go into PROGRAM mode;
    - *r89(ERR)* - ON state of this register will signal for alarm state in the user program;
    - *r90(CON)* - ON state shows that the *VPC* is connected to PC application via RS232C;
    - *r91(VPCIO)* - ON state shows that the *VPC* is connected to **VPCIO** expansion module;
    - *r92* - toggles between ON and OFF on each program cycle;
    - *r93* - 0.01 s ON-OFF cycle;
    - *r94* - 0.1 s ON-OFF cycle;
    - *r95* - 1 s ON-OFF cycle.

Each of these discrete registers (*i,o,t,c,r)* can serve as operand to the instructions of the *VPC* program language (see. ***Program instructions and mnemonics***). The end-of-curcuit type instructions *OUT, OUT NOT* and *LR* can take as operand only writable registers, i.e. outputs *o00-o29* or auxiliary registers *r00-r89*.

- analog inputs - *a00,...,a07*;
- analog outputs - *d00,...,d03*;
- auxiliary decimal registers - *p00,...,p77*:
  - *p00,...,p75* - common decimal registers with values *0000,...,9999.* their values can be modified by the user in **RUN & MONITOR** mode or transferred to/from the timer or counter values by the user program in **PROGRAM** mode;

  - *p76(parallel output*) - if *p76* is used in the user program, its value (mod 4096) is outputted binary by *o00-o11, o11* is MSB*;*

  - *p77(parallel input*) reads the decimal equivalent of *i00-i11,* where *i11* is the MSB.

Each of these decimal registers (*a,d,p*) can serve as an operand to the compare, transferring or arithmetic instruction of the *VPC* program language (see. ***Program instructions and mnemonics***). The transferring instructions *OUT* and *OUT** on these registers (except for *a00-a07* and *p77*) need a second operand, which serves as a source register of the transfer.

On power failure, all timers, all counters, *p00-p75* and *r72-r87* are saved.

IV. Program instructions and mnemonics

The instructions and the operands of the *VPC* PLCs are displayed on the LCD using a suitable mnemonics with the following meaning:

- "*END*" - end of program;
- *"LD"/"LD NOT"*^ (LoaD/LoaD NOT) - start of a new block/circuit with a NO/NC contact in the terms of the relay-contactor analogy, (the operand can be any discrete register or a decimal compare of two decimal operands);
- *"AND"/"AND NOT"* ^- a new NO/NC contact in series to the previous block, (the operand can be any discrete register or a decimal compare of two decimal operands);
- *"OR"/"OR NOT"* ^- a new NO/NC contact in parallel to the previous block, (the operand can be any discrete register or a decimal compare of two decimal operands);
- *"OUT"/"OUT NOT"*^, where NOT means inverse value:
  - outputs the circuit to output line, when the operand is a discrete register;
  - transfers the result of the transferring instruction to the destination operand, if the current value is ON (for *OUT*), or OFF (for *OUT*=OUT NOT*):

    OUT[*] destination,source results to the value of *source*

    OUT[*] destination+source results to the value of *destination+source*

    OUT[*] destination-source results to the value of *destination-source*
- "*LR*" (Latching Relay) - latching relay of Set-Reset type. The output is set and reset on two different circuits- the last one, and the last but one. The reset circuit has the priority;
- "*TIM00 - TIM31"* (Timer) - timer output. The output is set to ON after the timer value has elapsed with input ON. Any drop of the input value to OFF will reset the timer. *TIM00-TIM15* flow down and are reset to timer value, where *TIM16-TIM31* flow up and are reset to zero;
- *"CNT00 - CNT31"* (Counter) - counter output. The last circuit value resets the counter, the last but one is the counting input. The counter output is set to ON after the counter number of impulses have been counted on the input circuit.*CNT00-CNT15* count down and are reset to the count value, where *CNT16-CNT31* count up and are reset to zero;
- "*IL*" (InterLock) - beginning of a new branch. All subsequent *LD/LD NOT* will be depend on this interlock condition;
- "*ILC*" (InterLock Clear) - end of a branch;
- "*JMP*" (Jump) - conditional jump. The instructions up to the next "*JPE*" are performed only if the current condition is ON;
- "*JPE*" (Jump end) -end of the conditional jump;
- "*AND LD*" (AND Load) - the last block is AND-ed to the previous one;
- "*OR LD*" (OR Load) - the last two blocks are in parallel;
- *"??? ???"* - unknown instruction.

Examples of how to use these instruction are provided in **ChapterVII.**

^*NOT=*, when the operands are decimal: LD* is short for LD NOT.*

V. VPC modes

The *VPC* controllers work in two main modes: running mode with monitoring (**RUN & MONITOR**) and editing mode (**PROGRAM**) with search capabilities for output circuits.

In running mode the *VPC* performs the user program and thus controls the elements, connected to its inputs and outputs.

The editing mode is used only by qualified in PLC programming specialists for small and quick changes in the user program. Normally, the full-length initial program would be created on a PC using *VPC Host Interface*.

Toggling between the *VPC* modes is achieved by pressing the [**Esc**] button several times, until the LCD displays a message saying that the toggling can be done by pressing the "Ins"button:

"***To switch to RUN press <INS> key.***" or "***<INS> stops RUN <DEL> sets PASS***"

Switching from **PROGRAM** to **RUN & MONITOR** is allowed only if the user program is correct, otherwise the *VPC* will display the program error message (see ***Program errors***) and will remain in **PROGRAM** mode.

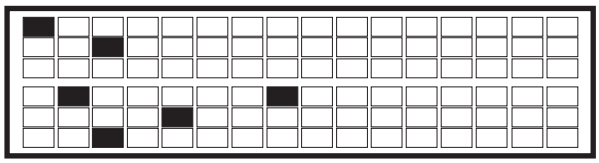
The controllers can be connected via the RS232C port to PC with *VPC Host Interface* running to transfer the user program in **PROGRAM** mode, or change and monitor the program registers in **RUN** mode.

V.a. RUN & MONITOR mode

While in **RUN & MONITOR** mode, the *VPC* PLC performs in successive order the instructions of the user program, starting from the instruction at address "*0000*", until the "*END*" instruction is reached. Then the program cycle repeats. This mode has four different display windows to indicate the state of the controller. Using them, the user can monitor the registers and the program flow of the controller. The navigation through the windows is done by the **[Esc]** button:

- **First window in RUN & MONITOR mode.**

This window displays the inputs and the outputs of the *VPC* controller. The upper part of the display is for the outputs, shown by 16 per line. The lower part of the display is for the inputs, shown in the same manner. A filled field represents the state ON, the empty one - OFF. On the following figure, the inputs *i01, i07*, *i20, i34* and the outputs *o00, o18* are ON, while all other inputs and outputs are OFF:



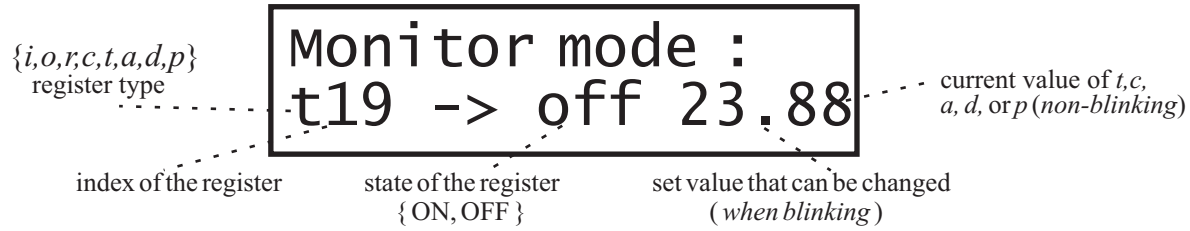
Outputs: *o00* ... *o15*
o16 ... *o29*

Inputs: *i00* ... *i15*
i16 ... *i31*
i32 ... *i39*

first window in ***RUN&MONITOR***

- **Second window in RUN & MONITOR mode.**

This window is used to monitor the state of the VPC registers. The [↵] button navigates through the type of the register, its index and its set value (for timers, counters and decimal registers). The chosen element blinks and can be changed by the arrow buttons [←] and [→]. In this fashion the user can select any of the *VPC* registers. The current state of the selected register is displayed right next to it. For timers, counters and the decimal registers (*a*, *d*, *p*), the current value is also displayed. The set value for those registers can be changed also:



- **Third window in RUN & MONITOR mode.**

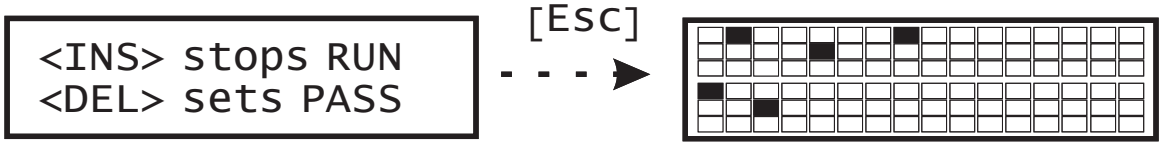
This windows is the editing mode of the *VPC* controller (see ***Editing mode PROGRAM***) without the ability to modify the user program. The first line of the display shows the instruction at the selected address. The last three characters of the second line show the state of the operand of the instruction shown:

scroll through the program using the buttons[↑] and [□] or

search circuit ends (see **SEARCH** in *Editing mode* ***PROGRAM***)

- **Fourth window in RUN & MONITOR mode.**

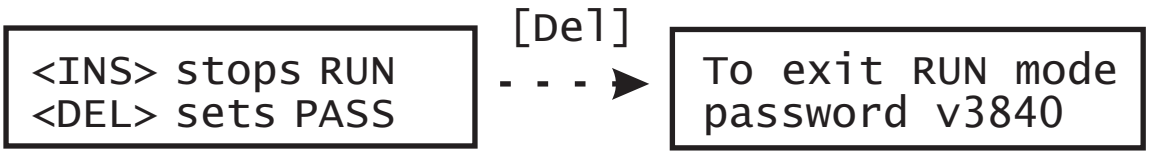
This is the last window of the **RUN & MONITOR** mode. Its purpose is to provide exit from this mode by pressing the [**Ins**] button, or set a password by the [**Del**] button. From this window the user can go back to the first window in **RUN & MONITOR** mode by pressing the [**Esc**] button:



last window in ***RUN & MONITOR***

first window in ***RUN & MONITOR***

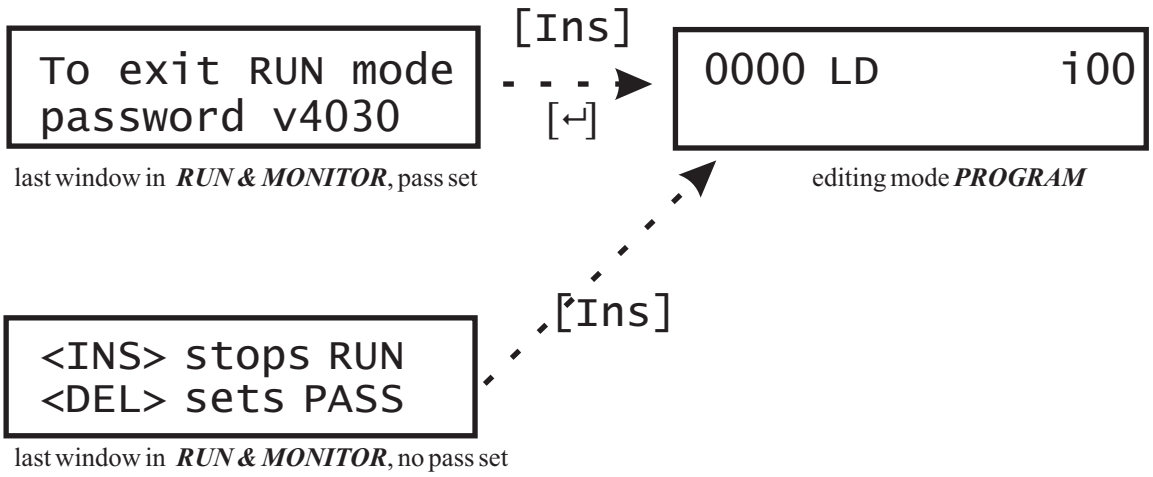
The user can enable the password protection that will prevent switching to **PROGRAM** mode and a possible change to the user program. With password set, the user will have to type the password in order to leave the **RUN & MONITOR** mode:



last window in ***RUN & MONITOR***, no pass set

last window in ***RUN & MONITOR***, pass set

The password for each *VPC* controller is the number after "VPC" in its name, i.e. the password for ***VPC3224A63*** is 3224.



last window in ***RUN & MONITOR***, pass set

editing mode ***PROGRAM***

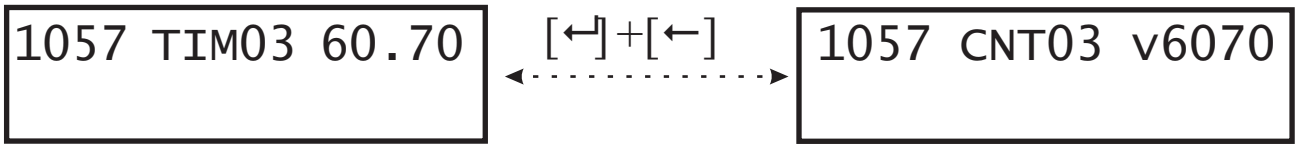
last window in ***RUN & MONITOR***, no pass set

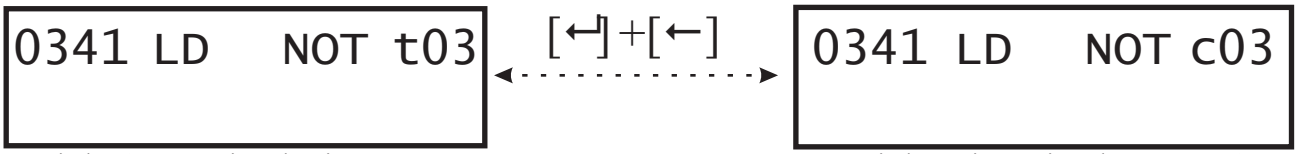
V.b. Editing mode PROGRAM

This mode is used to modify the user program via the inbuilt console of the *VPC* controllers. The program lines are shown on the display in the following format: address, instruction mnemonics, instruction operand(s), if any:

The user can scroll through the program using the buttons [↑] and [↓]. The arrow buttons are multi-speed which allows for a swift access to any address of the program. Pressing and holding [↑] and then pressing [←] will get the user to the beginning of the program at address *0000.* In the same manner, the combination of [↓] and [←] displays the end of the program, i.e. the first "*END*" instruction after the beginning. The current program line can be modified using the [↵] button. The blinking component of the instruction (mnemonics, operand type, operand index) can be modified by [←] and [→] buttons. The new value is saved by [↵] and the next component of the instruction will blink. After the last component is modified, the blinking cursor will hide. The user program must always end with the "*END*" instruction.

When using timers and counters as operand, one should keep in mind that each of the numbers *00,...,31* is assigned either to a timer, or to a counter. Reassigning an index from a timer to a counter can be done by getting to any program line where it is used, and then press and hold [↵] and press [←] :

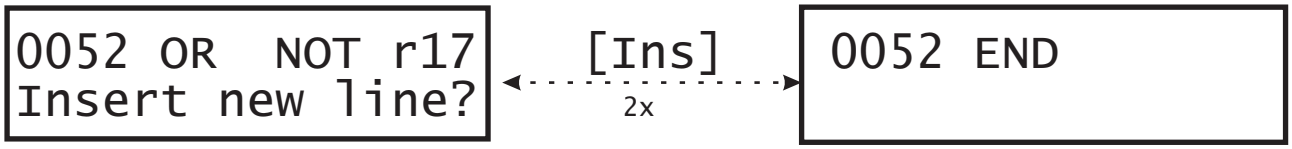




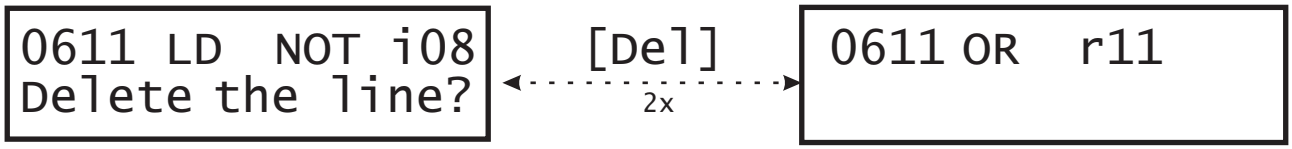
index *03* was assigned to timer *t03*

index *03* is reassigned to counter *c03*

Inserting and deleting program lines is done by the [**Ins**] and [**Del**] buttons. Confirmation of such an operation will be expected. [**Esc**] will cancel the operation, pressing again the chosen button will confirm the insertion or deletion. The second line of the display will show "*Inserting line..*" or "*Deleting line...*" until the operation is performing. The insertion will put a new "*END*" instruction at the current address:

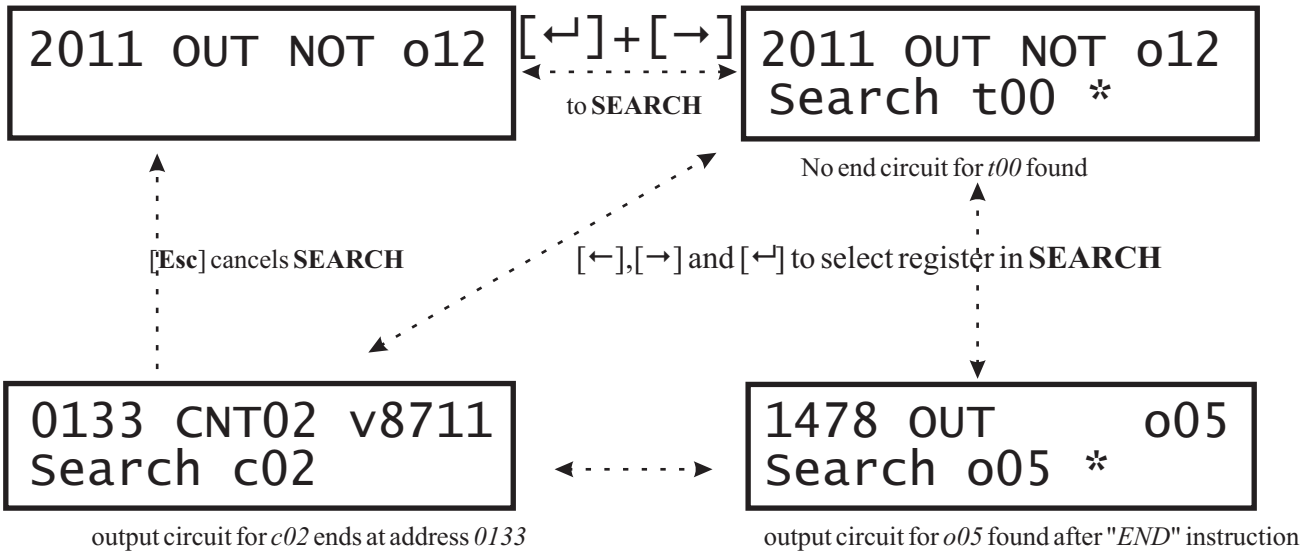


insertion of "*END*" instruction at address *0052*, the "*OR NOT r17*" instruction now goes to address *0053*



deletion of the instruction at address *0611*, its place will take the instruction that was at address *0612*

The editing mode **PROGRAM** has a **SEARCH** option for finding circuit ends in the user program. This option is turned on by pressing and holding [↵] and then pressing [→] . Pressing [**Esc**] will turn it off. With this option activated, the second line of the display will show "*Search t00 **". Using [↵], [←] and [→], the user can choose the desired register - output, auxiliary discrete register or decimal register. During the process of selecting the register, the first line of the display will show the program address with the instruction, where the circuit for the selected register ends. If there is no output circuit for the selected register in the user program's scope, then the mark '*' appears on the second line of the display:



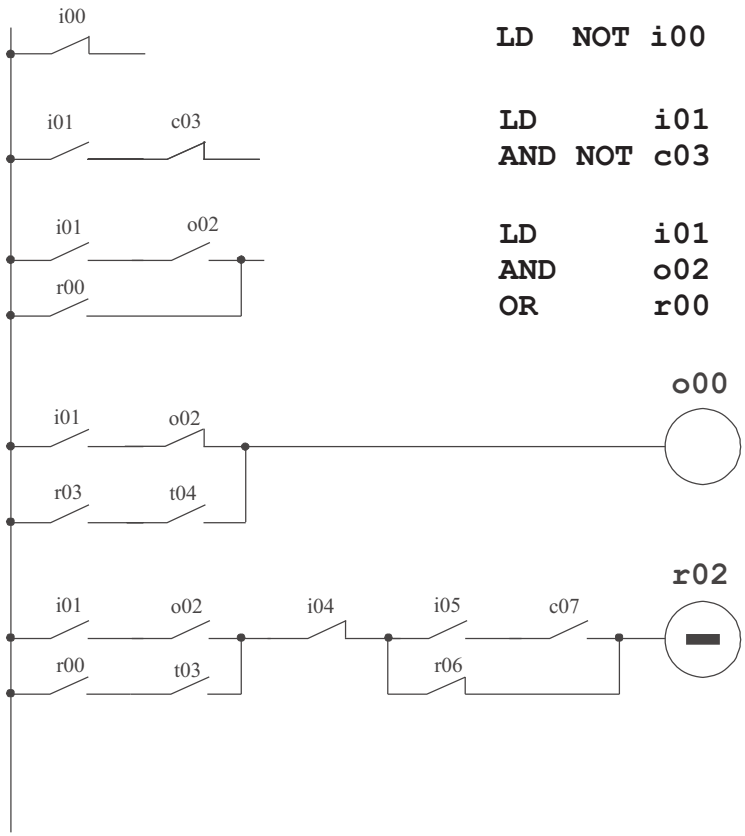
VI. Program errors

The *VPC* controller will check the user program for program errors each time the user presses [**Esc]** button. If there is no error, then the display will read "*To switch to RUN press <INS> key.*" and the user can switch to **RUN & MONITOR** mode. If an error is found, then the *VPC* controller remains in **PROGRAM** mode and displays the first error line with its address on the first line of the display. The second line of the display will show one of the following error messages:

- "*Oper duplication*" - more that one output circuit for a register;
- "*Invalid code!*" - unknown instruction;
- "*Invalid operand!*" - the operand is not recognized for the current instruction;
- "*Invalid value!*" - the timer/ counter value or a register index is not correct;
- "*'END' missing!*" - no "*END*" instruction found;
- "*IL (JMP) missing*" - the "*ILC*"/"*JPE*" instruction has no previous "*IL*"/"*JMP*" match;
- "*IL-JMP interlace*" - incorrect interlace between "*IL*"-"*ILC*" and "*JMP*"-"*JPE*" blocks.

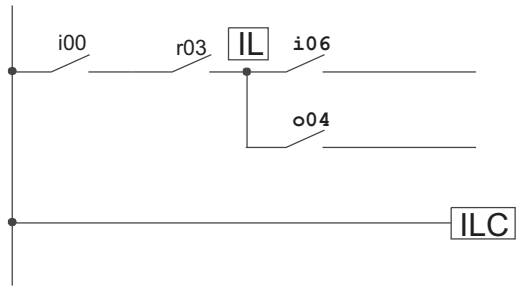
VII. Programming examples with relay ladder analogy

LD , LD NOT; *AND, AND NOT*; *OR, OR NOT*; *OR LD*; *AND LD*; *OUT, OUT NOT*:



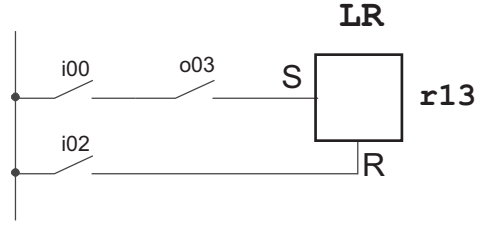

```
LD   NOT i00

LD       i01
AND NOT  c03

LD       i01
AND      o02
OR       r00

LD       i01
AND NOT  o02
LD       r03
AND      t04
OR LD
OUT      o00

LD       i01
AND      o02
LD       r00
AND      t03
OR LD
AND NOT  i04
LD       i05
AND      c07
OR  NOT  r06
AND LD
OUT NOT  r02
```

Block instructions *IL* (INTERLOCK) and *ILC* (INTERLOCK CLEAR) for beginning and ending of branches:



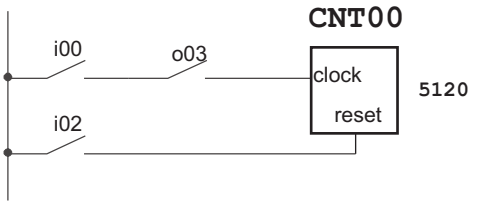

```
LD      i00
AND     r03
IL
LD      i06
...
LD      o04
...
ILC
```

Instruction *LR* (Latching Relay) is equivalent to a RS trigger:



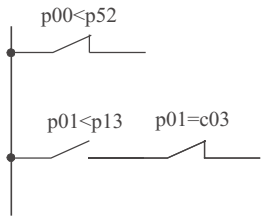

```
LD      i00
AND     o03
LD      i02
LR      r13
```

Instruction *CNT* for setting up a counter:



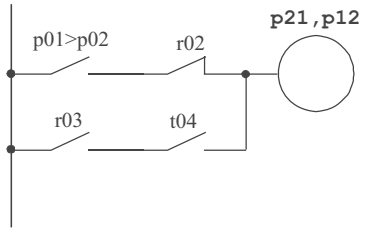

```
LD       i00
AND      o03
LD       i02
CNT00 v5120
```

Two-operand instructions for decimal compare *LD*; *LD **;*AND*; *AND**; *OR*; *OR **:




```
LD *p00<p52

LD  p01<p13
AND*p01=c03
```

Transferring value between registers based on a condition. The value of (*p12*) will be moved to (*p21*) using the *OUT* or *OUT**:




```
LD  p01>p02
AND NOT r02
LD      r03
AND     t04
OR LD
OUT p21,p12
```

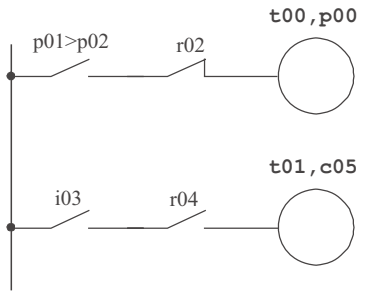
The actual transfer of the value will take place if the current state is ON (for the *OUT* instruction), or OFF (for the *OUT**, respectively). On the example above, the moving from *p12* to *p21* will be performed if any of the two branches is fulfilled.

The source of the transfer is the second operand, and the destination is the first one.

When the first operand is a timer or a counter, then the transfer will be to its **set value**.

If the second operand is a timer or a counter, then its **current** time/count **value** will be taken as a source.

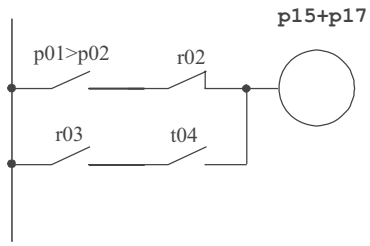
Example:



t00 will have the value of *p00* as a new **set value**

the **current count value** of *c05* will be the new **set value** of *t01*

Arithmetic addition and subtraction using the *OUT* and *OUT** instructions:

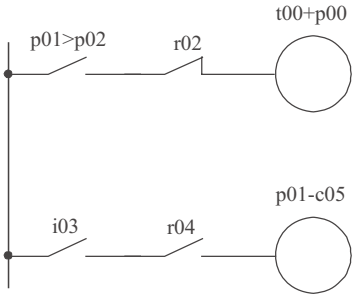



```
LD  p01>p02
AND NOT r02
LD      r03
AND     t04
OR LD
OUT p15+p17
```

The value of the second operand will be added to the value of the first operand and saved as a value of the first operand, if the current state is ON (for the *OUT* instruction), or OFF (for the *OUT** respectively). In the example above (*p15*):=(*p15*)+(*p17*).

If the first operand is a timer or a counter, then the addition will take place on its **set value**. When the second operand is a timer or a counter, then it will participate in the sum with its **current value**.

Examples:




```
LD  p01>p02
AND NOT r02
OUT t00+p00
```

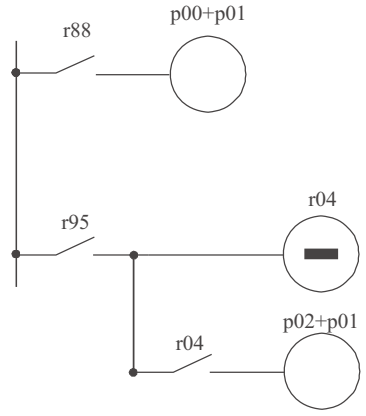
the **set value** of *t00* will increase with the value of *p00*

```
LD      i03
AND     r04
OUT t01-c05
```

p01 will increase with the **current** count **value** of *c05*

Mind that the arithmetic operation will take place ***as long as the condition*** is ON. The user should provide precise conditions when only a single addition or subtraction is desired.

Examples:




```
LD      r88
OUT p00+p01
```

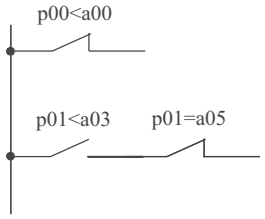
p00 will increase with *p01* on each program cycle

```
LD      r95
IL
OUT NOT r04
LD      r04
OUT p02+p01
ILC
```

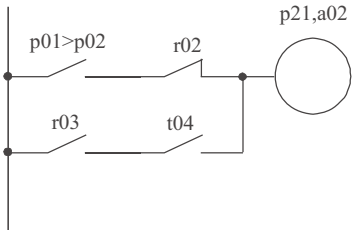
p02 will increase with *p01* on each second

Analog input and output registers:

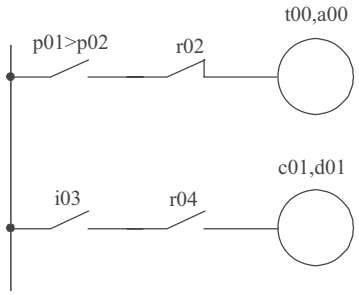
The analog inputs and outputs are treated the same way as the internal decimal registers. The only restriction is that the analog inputs cannot be destinations of the transferring and arithmetic instructions (*OUT* and *OUT**).




```
LD *p00<a00

LD  p01<a03
AND*p01=a05
```



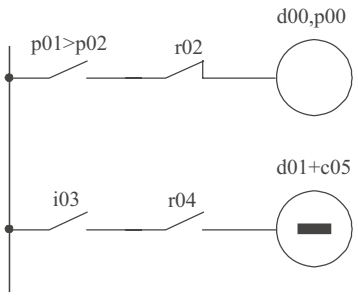

```
LD  p01>p02
AND NOT r02
LD      r03
AND     t04
OR LD
OUT p21,a02
```



the set value of *t00* will be the value of analog input *a00*

the set value of *c01* will be the value of analog output *d01*

Instructions with analog outputs as destination:



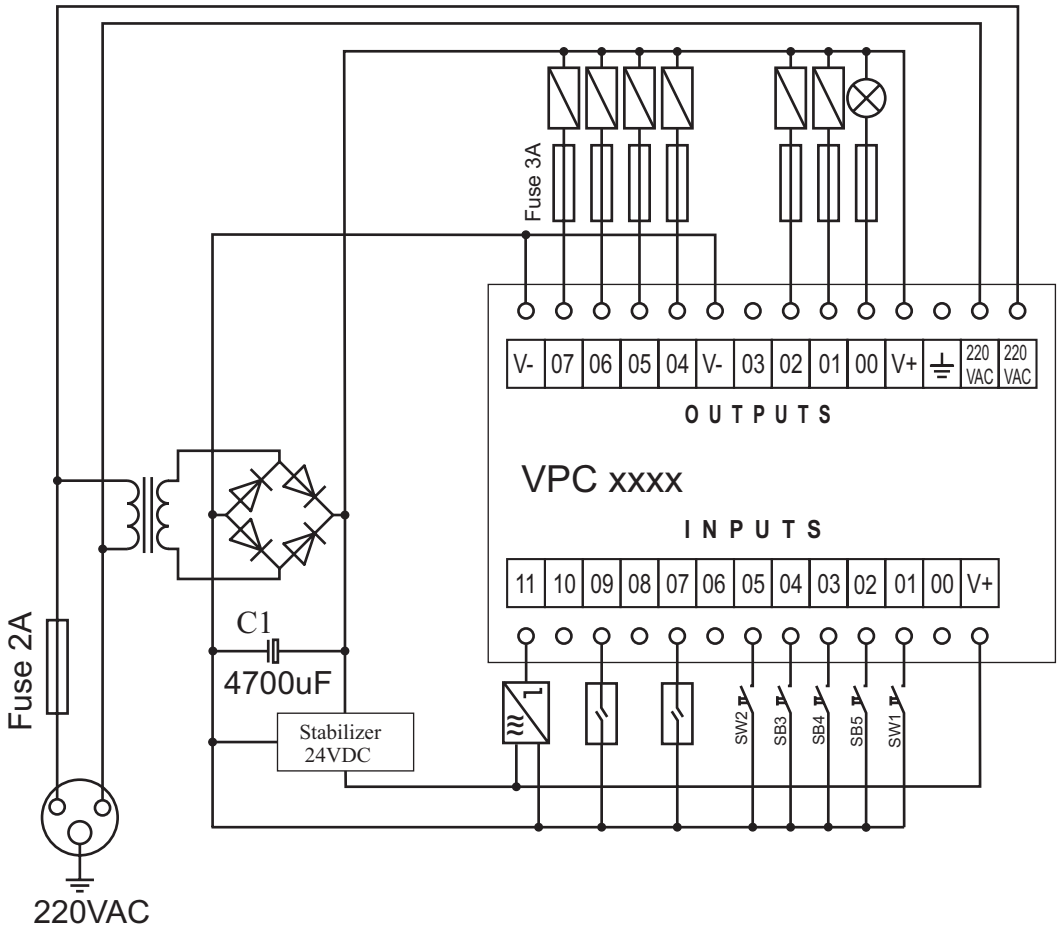

```
LD  p01>p02
AND NOT r02
OUT d00,p00

LD      i03
AND     r04
OUT*d01+c05
```

the analog output *d00* will take the value of *p00* when *p01>p02* and the state of *r02* is OFF

when any of the discrete inputs *i03* and *i04* drops to OFF state, the analog output *d01* will increase its value with the current value of the counter *c05*

VIII. VPC pinouts and wiring

The VPC inputs and outputs are NPN. This means that the active level is 0V.



Example of wiring sensors and buttons to the inputs and electromagnetic valves and other loads to the outputs of the *VPC* controller

It is recommended that the rectified voltage is stabilized at 24VDC. The common V+ input is tied to the stabilized source, and the common V+ output is tied to the non-stabilized source.

All *VPC* controllers have transistor outputs that can be directly wired to loads of active or inductive character, but not greater than 2 ADC. It is recommended that fast fuses are set between the outputs and the loads to avoid damage on possible short circuits.