PLC CONTROLLERS

Series
Nexus/Lynx/Vega/PC100

PLC Operation

elap ELAP VIA VITTORIO VENETO, 4 • I–20094 CORSICO (MI) • TEL. +39.02.451956
FAX +39.02.45103406 • E–MAIL INFO@ELAP.IT • SITE WWW.ELAP.IT

PLC INSTRUCTIONS FOR PC100N / NEXUS / LYNX / VEGA

All writing instructions and information are conditioned by the logical chain preceding the function, except for the following functions: end, endp, ret, nop.

Operands

#	= constant value on decimal base
#H	= constant value on esadecimal base
I	= input
O	= output
M	= inside relays, special memories, axes memories.
T	= timer
C	= counter
R	= shift register
S	= step (for PC100N, Nexus, Lynx only)
B	= byte in the data register area
W	= word in the data register area
D	= double-word in the data register area
AB	= byte absolute address in the basic memory
AW	= word absolute address in the basic memory
FB	= byte in the far area
FW	= word in the far area
FD	= double-word in the far area

gAn = axis parameter

special codes: g A n where g = board or group index
1 to 5 for Pc100N and 1 to 2 for Nexus, Lynx, Vega
A = fix
n = 3-register address (000 to 216)

I/O General Instructions

ld	i, o, m, t, c, r, s	-| |-
ldc	i, o, m, t, c, r, s	-|/|-
and	i, o, m, t, c, r, s	-| |-
andc	i, o, m, t, c, r, s	-|/|-
or	i, o, m, t, c, r, s	-| |-
orc	i, o, m, t, c, r, s	-|/|-
xor	i, o, m, t, c, r, s	
xnor	i, o, m, t, c, r, s	
sto	o, m	-( )-
set	o, m	-(S)-
res	o, m	-(R)-

andld (complex chains)
orld

Fronts

eg. ld i0.0

```
re      (positive front)        -[re]-          and    i0.1
fe      (negative front)        -[fe]-          re
                                                sto    o0.0
```

Total:128 max.

Timer

```
tset    n,t                                     n = Timer number (0,127)
                                                t = #, w (16-bit time value)
                                                eg.  ld     i0.0
                                                     tset   1,#50   ;set timer1 = 0.50s

                                                       ---[T  ]---
                                                           t1
                                                          #50
```

Counter

```
set counter up:                    -[C-UP]--[   ]-          eg. counter up with preset = 100 e
cuset   n, preset, load              Cx     load            starting value  = 0
                                     #100   #0              ld     i0.0    ; bit di reset/load
set counter down:                    (ck) --[   ]-          ld     i0.1    ; clock bit
cdset   n, preset, load                              cuset  1,#100,#0
```

where   n = counter index from 0 to 127
        preset = count to be reached at 16 bit (#, w)
        load = starting value at 16 bit (#, w)

Step Register (for PC100N, Nexus, Lynx only)

```
sset    n,#nbit  (configuration)   where   n = step register from 0 to 15
                                           nbit = register dimension in bit (max 128)

spr sn,#nbit     (register reset)  where   sn = step register from s0 to s15
                                           nbit = starting bit after the reset
                                           eg.: spr s0,#5

spr sn     (register reset)   where sn = step register from s0 to s15
                              total reset and start at the 1st bit

step     bit      (development)
                            eg.    ld       i0.0      ; configurates the step-register
   --| |-----[STEP]-               sset     0,#20     ; uses 20 bit from s0.0 to s0.19
     i0.0      s0                  ...
               #20                 ...
  ...                              ld       m0.0      ; development
  ...                              step     s0.0
                                   ld       m1.0
   --| |--------( )-               step     s0.1
   m0.0                            s0.0     ...
                                   ld       m19.0
  --| |--------( )-                step     s0.19
   m1.0                            s0.1
  ....                              ...
  --| |--------( )-                 ld           i0.0
   i0.0      spr s0       spr       s0
```

Shift Register

```
rset    n,#nbit  (Datum)         -[SFR]-    eg.   ld     i0.0    ;Datum
                                  R0              ld     i0.1    ;Reset
                                  #8              ld     i0.2    ;Clock
                 (reset)         -[   ]-          sfr    0,#8     :reg.0, 8 bit
                 (clock)         -[   ]-
```

where n = shift register index from 0 to 15
nbit = bit number of the shift register (8,16,24,32,...256; multiple of 8)

Arithmetical Operations

```
addi    op1,op2,op3                 where   op1 = 1st operand (#, w, d)
addd    op1,op2,op3                         op2 = 2nd operand (#, w, d)
addr    op1,op2,op3                         op3 = result (w, d)

subi    op1,op2,op3                 the suffix
subd    op1,op2,op3                 i indicates an operation between integers (16 bits)
subr    op1,op2,op3                 d indicates an operation between longs (32 bits)
                                    r sta indicates an operation between floats  (32 bits)
muli    op1,op2,op3
muld    op1,op2,op3                 eg..   ld    i0.0    ;enabling sum
mulr    op1,op2,op3                        addi  w0,#1,w0

divi    op1,op2,op3                         -[  ]-[  ]-[  ]-[  ]-
divd    op1,op2,op3                 addi op1  op2  op3
divr    op1,op2,op3                         w0   #1   w0

inc     op1                         Increment/Decrease
dec     op1                         op1 = 16-bit operand (w)
```

Conversion operations by factor (for NEXUS, LYNX, VEGA only)

```
xfatt op1,op2,op3                   Performs op3=(op1*op2)/1000000
dfatt op1,op2,op3                   Performs op3=1000000/(op1*op2)

                                    op1 = source (d, @w, gAn)
                                    op2 = multiplier (d, #, gAn)
                                    op3 = destination (d, @w, gAn)
```

Boolean operations (16-bit operands only)

```
andw   op1,op2,op3                  where   op1 = 1st operand (#, Aw, w)
orw    op1,op2,op3                          op2 = 2nd operand (#, Aw, w)
xorw   op1,op2,op3                          op3 = result (Aw, w)
not    op1,op2

eg.    -[  ]-[  ]-[  ]-[  ]-         ld    i0.0    ;enabling AND
      andw op1  op2  op3             andw  w0,#10h,w2
               w0  #10h  w2
```

Comparison operations between integers

```
 =i  op1,op2,op3      equal             where op1,op2 = operand (#, w)
<>i  op1,op2,op3      different                   op3 = result oX.X, mX.X
 >i  op1,op2,op3      more than
 <i  op1,op2,op3      less than         eg.        ld    i0.0    ;enables the comparison
                                                  >i     w0, #100, o0.3

                                                  -[  ]-[  ]-[  ]—[ ]-
                                                  >i   op1  op2  op3
                                                  w0   #100 o0.3
```

Comparison operations between longs

```
 =d  op1,op2,op3      equal             where op1,op2 = operand (#, d)
<>d  op1,op2,op3      different                   op3 = result oX.X, mX.X
 >d  op1,op2,op3      more than
 <d  op1,op2,op3      less than         eg.       ld    i0.0    ;enables the comparison
                                                  >i    d0, #100, o0.3

                                                  -[  ]-[  ]-[  ]—[ ]-
                                                  >d   op1  op2  op3
                                                  d0   #100 o0.3
```

Comparison operations between floats

```
 =r  op1,op2,op3      equals      where op1,op2 = operand (#, d)
<>r  op1,op2,op3      different         op3 = result oX.X, mX.X
 >r  op1,op2,op3      more than
 <r  op1,op2,op3      less than         eg.  ld    i0.0    ;enables the comparison
                                            >d    d0, #100, o0.3

                                             -[  ]-[  ]-[  ]—[ ]-
                                             >d   op1  op2  op3
                                             d0   #100 o0.3
```

REMARK: the operand 3 in all comparison instructions is enabled by an instruction equivalent to **sto**.

Jumps

```
jmp    label                            eg.  ld    i0.0    ;enables the jump
                    -[JMP]-                  jmp   label1
                    label1
```

Conversion operations

```
itod   op1,op2   (int to long)        op1 = value to be converted (#, w, d)
```

dtoi op1,op2 (long to int) op2 = converted value (w, d)
dtor op1,op2 (long to float)
rtod op1,op2 (float to long) eg. ld i0.0 ;Enables the conversion
dtor #100,d0

```
-[   ]--[   ]--[   ]-
  dtor   op1    op2
 #100   d0
```

bcdbin	op1,op2	(BCD to BIN)	op1 = value to be converted (#, Aw, w)
binbcd	op1,op2	(BIN to BCD)	op2 = converted value (Aw, w)

Shift operations (16-bit operands)

shl	op1,#nbit	(shift left)	op1 = value to be shifted (Aw, w)
shr	op1,#nbit	(shift right)	nbit = bit number to be shifted (1 to 16)

Rotation operations with carry (16-bit operands)

rol	op1,#nbit	(rotate left)	op1 = value to be shifted (Aw, w)
ror	op1,#nbit	(rotate right)	nbit = number of bits to be rotated (1 to 16)

Copy and allocation operations

movb	op1,op2	op1 = #, Ab, b, @w	op2 = Ab, b, @w
movw	op1,op2	op1 = #, Aw, w, @w	op2 = Aw, w, @w
movd	op1,op2	op1 = #, d, @w	op2 = d, @w
movr	op1,op2	op1 = #	op2 = d, @w (assigns a real value)

movn op1,op2,#nbit op1, op2 = Ab, start-bit nbit from 1 to 128

setn start-bit,#nbit,#value where start-bit = starting bit address
nbit = number of bits to set/reset
value = 0, 1

setb start-byte,#nbyte,#value where start-byte = start-byte address (Ab)
nbyte = number of bytes to be overwritten
value = 0 -:- 255

Copy operations to FAR (For NEXUS, LYNX, VEGA only)

movfb	op1,op2	op1 = b, fb, @w, @d	op2 = b, fb, @w, @d
movfw	op1,op2	op1 = w, fb, @w, @d	op2 = w, fw, @w, @d
movfd	op1,op2	op1 = d, fd, @w, @d	op2 = d, fd, @w, @d

copyfb s, d, #n s = data source: b, fb
d = data destination: b, fb
#n = number of bytes to be copied (1 to 4096)

Copy operations to data structures (For NEXUS, LYNX, VEGA only)

Movstrc #Nstructure,Index,Member,ds Copies a member of the structure to the far area
to the address ds near
Index = #, w

Member = #, #0 (copies the whole structure)
Ds = b, w, d destination address
If the member is = 0 the destination address must be d only.

Movstrc sC,#Nstructure,Index,Member — Copies a near datum in a member of a far structure
sg = b, w, d source address
Index = #, w
Member = #,#0 (copies the whole structure)
If the member is = 0 the source address must be d only.

Further instructions

End — It determines the end of the main program. After this function, possible further subroutines must be defined.

Endp — It determines the end of the area where functions are defined.
This instruction must always be entered after the instruction End even when no other function is required.

Eg. Main Program
End
Func1
Func2
Endp

Function Call

Call nome

nome = name of the function (max 6 characters)
REMARK: The Call function must be defined after the instruction END.

Ret — Return after a function
REMARK: Any function must end with Ret

Interrupts

Int n — n = 0,1,2,3
Selects one of the four interrupt channels

Int all — Selects all interrupt channels

Ei n — n = 0,1,2,3
Enables one of the four interrupt channels

Ei all — Enables all interrupt channels

Di n — n = 0,1,2,3
Disables one of the four interrupt channels

Di all — Disables all interrupt channels

Nop — No operations. The microprocessor executes an instruction nop.

`RTask` nome — Call of a function performed in multitasking
in asynchronous way in respect to the PLC scanning.
Nome = name of the function (max 6 characters)
REMARK: The call function must be defined after the instruction END
and ended by the instruction RET.

PLC Serial Protocol

In the following paragraphs the identifiers ‘s’ and ‘d’ are referred to respectively as ‘source’ and ‘destination’. The second column indicates which data type can be used as source and destination.

String Constant

label,"sssssssssss"
Max. length: 255 characters with last character "null" added by the compiler.
. Total number of characters: max. 2048.

Instruction 'convert number -> string with sign'

binasc s, d
s = source of the numerical value b, w, d
d = string start byte b, (10 bytes fix).
Eg. binasc w50, b20

Instruction'convert string with sign -> number'

ascbin s, d
s = string start byte b
d = numerical value in double word d
Eg. ascbin b34, d7

Instruction for the string chaining

strcpy s, d, n
s = label (constant string)
d = string start destination byte b
n = number of characters to be copied
n must be in the range 0 to 255
if = 0 copies the whole string
Eg. strcpy frase3, b10, #45

Instruction for the string transmission

copyb s, d, n
s = source buffer start byte b
d = BTX buffer specialized for the transmission
n = number of characters to be copied
n must be in the range 1 to 255

This instruction must be conditioned by a bit defined in the special function register, indicating that the transmission buffer id empty.

Instruction 'receive string'

copyb s, d, #
s = BRX buffer specialized for the receipt
d = destination buffer start byte b
# = number of characters to be copied
# must be in the range 1 to 255

This instruction must be conditioned by a bit defined in the special function register, indicating the correct receipt of the number of characters specified by the instruction setbrx.

Instruction 'Receive number of characters'

setbrx c,#
c = Character to be releaved in the receipt, which generates an interrupt
# = number of characters to be received
# must be in the range 1 to 255

If c = 0 an interrupt is generated after the receipt of # characters.
The recognition of the character c must place the receipt pointer on the first BRX byte

Instruction 'Reset BRS Buffer'

clrbrx

Instruction 'Set the serial line'

setcom B, p, b, s	B = baud rate (1200, 2400, 4800, 9600, 19200, 38400)
	p = parity (none, even, odd)
	b = bit (7, 8)
	s = stop bit (1, 2)

Instruction 'Link station'

linkrd #id, s, d, #	#id = Identifying code of the station to be read
	s = source of the station to be read (see the PLC memory map)
	d = destination
	# = Word number to be read (max 16)
linkwr #id, s, d, #	#id = Identifying code of the station to be written
	s = source
	d = destination of the station to be written (see the PLC memory map
	# = Word number to be read (max 16)
linkid #	# = Station identification

REMARK:
linkwr, linkrd set the station as **MASTER.**
linkid sets the station as **SLAVE.**

AXES Addressing (axes 3 and 4 available for Nexus only)

The following paragraphs conventionally refer to the axes as:

1	group 1	axis 0
2	group 1	axis 1
3	group 2	axis 0
4	group 2	axis 1

Axes state signalling bit

Axes	1	2	3	4	
	i1.0	i1.8	i2.0	i2.8	direction
	i1.1	i1.9	i2.1	i2.9	index not found
	i1.2	i1.10	i2.2	i2.10	in position
	i1.3	i1.11	i2.3	i2.11	pre-signal
	i1.4	i1.12	i2.4	i2.12	positive limit switch
	i1.5	i1.13	i2.5	i2.13	negative limit switch
	i1.6	i1.14	i2.6	i2.14	index found
	i1.7	i1.15	i2.7	i2.15	following error

Matching of the axes output bits and NEXUS, LYNX, VEGA profile

Axes	1	2	3	4	
	o1.0	o1.8	o2.0	o2.8	Position register load
	o1.1	o1.9	o2.1	o2.9	index following micro
	o1.2	o1.10	o2.2	o2.10	position register reset
	o1.3	o1.11	o2.3	o2.11	axis start
	o1.4	o1.12	o2.4	o2.12	analog output enabling
	o1.5	o1.13	o2.5	o2.13	manual forward
	o1.6	o1.14	o2.6	o2.14	manual backward
	o1.7	o1.15	o2.7	o2.15	find index

Matching of the axes input memories bits and NEXUS, LYNX, VEGA profile

Axes	1 - 2	3 - 4	
	m40.0, Ab144, Aw72	m41.0, Ab146, Aw73	Data loss
	m40.1	m41.1	Error position 0
	m40.2	m41.2	Error position 1

Matching of the axes output memories bits and NEXUS, LYNX, VEGA profile

Axes	1 - 2	3 - 4	
	m40.8, Ab145, Aw72	m41.8, Ab147, Aw73	Errors reset
	m40.9	m41.9	DAC 0 direct enabling
	m40.10	m41.10	DAC 1 direct enabling

Parameters for two axes for NEXUS, LYNX, VEGA

Type = b -> 1 byte, w -> word, d -> double word

N.Par	Type	Min.Value	Max.Value	Parameter
4	d	-8388608	8388607	Axis 0: Position
124	d	-8388608	8388607	Axis 1: Position
8	d	-8388608	8388607	Axis 0: Target position
12	d	-8388608	8388607	Axis 0: Position Load
32	w	1	9999	Axis 0: Tolerance range
24	d	-8388608	8388607	Axis 0: Dead band
34	w	1	100	Axis 0: % Speed
36	w	1	999	Axis 0: Accel./Decel. Time
38	w	1	999	Axis 0: Proportional gain
40	w	0	999	Axis 0: Derivative gain
42	w	0	999	Axis 0: Integral gain
44	w	0	32000	Axis 0: Following error
46	w	1	999	Axis 0: Following error output delay
48	w	1	999	Axis 0: Emergency deceleration time
50	w	1	999	Axis 0: Inspection time
52	w	1	99	Axis 0: % 1st Manual speed.
54	w	1	99	Axis 0: % 2nd Manual speed
56	w	1	999	Axis 0: Speed change time
60	w	1	999	Axis 0: In position output time
16	d	-8388608	8388607	Axis 0: Positive limit switch
20	d	-8388608	8388607	Axis 0: Negative limit switch
28	d	10	400000	Axis 0: Max.speed (pulse/sec)
62	w	1	99	Axis 0: % Limit switch find speed
64	w	1	99	Axis 0: % Zero index find speed
58	w	10	999	Axis 0: Zero index error timeout
66	w	-2048	2047	Axis 0: DAC direct output
68	w	0	255	Axis 0: Operating mode
70	w	0	6	Axis 0: Decimal digits
72	d	10000	10000000	Axis 0: Position factor pulses
76	d	-9999999	9999999	Axis 0: Target position (with factor)
80	d	-9999999	9999999	Axis 0: Load (with factor)
84	d	-9999999	9999999	Axis 0: Positive limit switch (with factor)
88	d	-9999999	9999999	Axis 0: Negative limit switch (with factor)
92	d	-9999999	9999999	Axis 0: Dead band (with factor)
96	w	1	9999	Axis 0: Tolerance range (with factor)
98	w	-32768	32767	Axis 0: Analog input
100	d	-9999999	9999999	Axis 0: Following error (reading only)

128	d	-8388608	8388607	Axis 1: Target position
132	d	-8388608	8388607	Axis 1: Position Load
152	w	1	9999	Axis 1: Tolerance range
144	d	-8388608	8388607	Axis 1: Dead band
154	w	1	100	Axis 1: % speed
156	w	1	999	Axis 1: Accel/decel. time
158	w	1	999	Axis 1: Proportional gain
160	w	0	999	Axis 1: Derivative gain
162	w	0	999	Axis 1: Integral gain
164	w	0	32000	Axis 1: Following error
166	w	1	999	Axis 1: Following error output time
168	w	1	999	Axis 1: Emergency deceleration time
170	w	1	999	Axis 1: Inspection time
172	w	1	99	Axis 1: % 1st Manual speed.
174	w	1	99	Axis 1: % 2nd Manual speed.
176	w	1	999	Axis 1: Speed change time
180	w	1	999	Axis 1: In positionoutput time
136	d	-8388608	8388607	Axis 1: Positive limit switch
140	d	-8388608	8388607	Axis 1: Negative limit switch
148	d	10	400000	Axis 1: Max. speed (pulse/sec)
182	w	1	99	Axis 1: % Limit switch find speed
184	w	1	99	Axis 1: % Zero index find speed
178	w	10	999	Axis 1: Zero index timeout error
186	w	-2048	2047	Axis 1: DAC direct output
188	w	0	255	Axis 1: Operating mode
190	w	0	6	Axis 1: Decimal digits
192	d	10000	10000000	Axis 1: Position factor pulses
196	d	-9999999	9999999	Axis 1: Target position (with factor)
200	d	-9999999	9999999	Axis 1: Load (with factor)
204	d	-9999999	9999999	Axis 1: Positive limit switch (with factor)
208	d	-9999999	9999999	Axis 1: Negative limit switch (with factor)
212	d	-9999999	9999999	Axis 1: Dead band(with factor)
216	w	1	9999	Axis 1: Tolerance range (with factor)
218	w	-32768	32767	Axis 1: Analog input
220	d	-9999999	9999999	Axis 1: Following error (reading only)

Axes operating mode - parameters 68 and 188

bit 0 = Load / Sum	(0 = load, 1 = sum)
bit 1 = 0 Index find direction	(0 = down, 1 = up)
bit 2 = 0 Index find mode	(0 = inside, 1 = outside)
bit 3 = Inverter mode	(0 = bipolar, 1 = unipolar)
bit 4 = Profile mode	(0 = trapezoidal, 1 = es)
bit 5 = Axis mode group 1 axis 1	(0 = normal, 1 = interpolated)
bit 6 = Axis mode group 2 axis 0	(0 = normal, 1 = interpolated)
bit 7 = Axis mode group 2 axis 1	(0 = normal, 1 = interpolated)

REMARK: Bits 5,6,7 are only valid if they were set in the parameter 68 axes group 1.

Special memories addressing

Memory bit matching in the special function register

m38.0, Ab140, Aw70	bit 0 of the identification code in link mode
m38.1	bit 1 of the identification code in link mode
m38.2	bit 2 of the identification code in link mode
m38.3	Buf. transmission (BTX) empty (empty = 1)
m38.4	Buf. receipt (BRX) available data (avail.data = 1)
m38.5	
m38.6	
m38.7	
m38.8, Ab141	
m38.9	
m38.10	
m38.11	
m38.12	Save counter values (save = 1)
m38.13	Save data bank 1 (w512 to w1023; save = 1)
m38.14	Save data bank 2 (w1024 to w153; save = 1)
m38.15	Save data bank 3 (w1536 to w2047; save = 1)

m39.0, Ab142, Aw71	PLC first scan (= 1 during the first PLC cycle only)
m39.1	Error ‘division by 0’ (error = 1)
m39.2	Carry of the shift instruction
m39.3	Oscillator at 10Hz (duty cycle 50%)
m39.4	Oscillator at 1 Hz (duty cycle 50%)
m39.5	
m39.6	
m39.7	
m39.8, Ab143	Bit always = 1
m39.9	Field voltage (Field Ok = 1)
m39.10	Battery voltage (battery Ok = 1) Ok =>2.75V.
m39.11	
m39.12	
m39.13	
m39.14	Dual-debug enabling (enabled = 1)
m39.15	WatchDog (intervention = 1)

List address bit / byte / word

Field fast inputs		Aw0	Ab0 : Ab1	i0.0:i0.3
Axes inputs	Axis1	Aw1	Ab2	i1.0 : i1.7
	Axis2		Ab3	i1.8 : i1.15
	Axis3	Aw2	Ab4	i2.0 : i2.7
	Axis4		Ab5	i2.8 : i2.15
Field inputs		Aw3	Ab6	i3.0 : i3.7
			Ab7	i3.8 : i3.15
		Aw4	Ab8	i4.0 : i4.7
			Ab9	i4.8 : i4.15
Axes outputs	Axis1	Aw17	Ab34	o1.0 : o1.7
	Axis2		Ab35	o1.8 : o1.15
	Axis3	Aw18	Ab36	o2.0 : o2.7
	Axis4		Ab37	o2.8 : o2.15
Field outputs		Aw21	Ab42	o5.0 : o5.7
			Ab43	o5.8 : o5.15
		Aw22	Ab44	o6.0 : o6.7
			Ab45	o6.8 : o6.15
Non-ritentive memories		Aw32	Ab64	m0.0 : m0.7
			Ab65	m0.8 : m0.15
		Aw33	Ab66	m1.0 : m1.7
			Ab67	m1.8 : m1.15
		Aw34	Ab68	m2.0 : m2.7
			Ab69	m2.8 : m2.15
		Aw35	Ab70	m3.0 : m3.7
			Ab71	m3.8 : m3.15
		Aw36	Ab72	m4.0 : m4.7
			Ab73	m4.8 : m4.15
		Aw37	Ab74	m5.0 : m5.7
			Ab75	m5.8 : m5.15
		Aw38	Ab76	m6.0 : m6.7
			Ab77	m6.8 : m6.15
		Aw39	Ab78	m7.0 : m7.7
			Ab79	m7.8 : m7.15
		Aw40	Ab80	m8.0 : m8.7
			Ab81	m8.8 : m8.15
		Aw41	Ab82	m9.0 : m9.7
			Ab83	m9.8 : m9.15
		Aw42	Ab84	m10.0 : m10.7
			Ab85	m10.8 : m10.15
		Aw43	Ab86	m11.0 : m11.7
			Ab87	m11.8 : m11.15
		Aw44	Ab88	m12.0 : m12.7
			Ab89	m12.8 : m12.15
		Aw45	Ab90	m13.0 : m13.7
			Ab91	m13.8 : m13.15
		Aw46	Ab92	m14.0 : m14.7
			Ab93	m14.8 : m14.15
		Aw47	Ab94	m15.0 : m15.7
			Ab95	m15.8 : m15.15

Retentive memories		**Aw48**	**Ab96**	**m16.0 : m16.7**
			Ab97	m16.8 : m16.15
		Aw49	Ab98	m17.0 : m17.7
			Ab99	m17.8 : m17.15
		Aw50	Ab100	m18.0 : m18.7
			Ab101	m18.8 : m18.15
		Aw51	Ab102	m19.0 : m19.7
			Ab103	m19.8 : m19.15
		Aw52	Ab104	m20.0 : m20.7
			Ab105	m20.8 : m20.15
		Aw53	Ab106	m21.0 : m21.7
			Ab107	m21.8 : m21.15
		Aw54	Ab108	m22.0 : m22.7
			Ab109	m22.8 : m22.15
		Aw55	Ab110	m23.0 : m23.7
			Ab111	m23.8 : m23.15
		Aw56	Ab112	m24.0 : m24.7
			Ab113	m24.8 : m24.15
		Aw57	Ab114	m25.0 : m25.7
			Ab115	m25.8 : m25.15
		Aw58	Ab116	m26.0 : m26.7
			Ab117	m26.8 : m26.15
		Aw59	Ab118	m27.0 : m27.7
			Ab119	m27.8 : m27.15
		Aw60	Ab120	m28.0 : m28.7
			Ab121	m28.8 : m28.15
		Aw61	Ab122	m29.0 : m29.7
			Ab123	m29.8 : m29.15
		Aw62	Ab124	m30.0 : m30.7
			Ab125	m30.8 : m30.15
		Aw63	Ab126	m31.0 : m31.7
			Ab127	m31.8 : m31.15
		Aw64	Ab128	m32.0 : m32.7
			Ab129	m32.8 : m32.15
		Aw65	Ab130	m33.0 : m33.7
			Ab131	m33.8 : m33.15
		Aw66	Ab132	m34.0 : m34.7
			Ab133	m34.8 : m34.15
		Aw67	Ab134	m35.0 : m35.7
			Ab135	m35.8 . m35.15
		Aw68	Ab136	m36.0 : m36.7
			Ab137	m36.8 : m36.15
		Aw69	Ab138	m37.0 : m37.7
			Ab139	m37.8 : m37.15
Check memories		Aw70	Ab140	m38.0 : m38.7
(Special function register)			Ab141	m38.8 : m38.15
		Aw71	Ab142	m39.0 : m39.7
			Ab143	m39.8 : m39.15
Axes memories	Axis1	Aw72	Ab144	m40.0 : m40.7
	Axis2		Ab145	m40.8 : m40.15
	Axis3	Aw73	Ab146	m41.0 : m41.7
	Axis4		Ab147	m41.8 : m41.15

PC 100N NEXUS LYNX VEGA

PLC INSTRUCTIONS FOR PC100N / NEXUS / LYNX / VEGA

PLC Serial Protocol

AXES Addressing

Special memories addressing

List address bit / byte / word