

KOSTAC PZ Manual







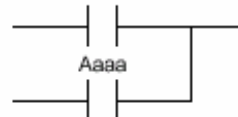
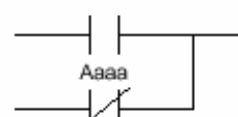
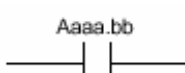
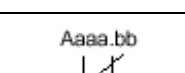
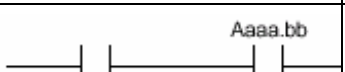
General Specification

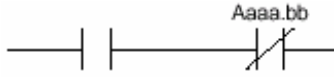
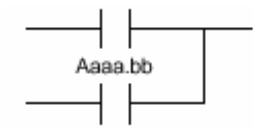
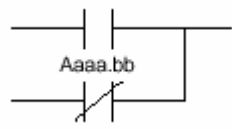
Items	Specification			
Input Voltage Range	DC22-24V (DC Type) AC85-264V 50 60Hz(AC Type)			
Operating Temperature	0°C to 55°C			
Storage Temperature	-20°C to 70°C			
Ambient Humidity	5% - 95% Relative Humidity (non-condensing)			
Environment	No Corrosive Gases			
Insulation Resistance	> 10M ohm at 500VDC			
Voltage Withstand (dielectric)	1 minute @ 500 VAC between primary, secondary, field ground (DC Type) 1 minute @ 1500 VAC between primary, secondary, field ground (AC Type)			
Vibration resistance	MIL STD 810C, Method 514.2			
Shock resistance	MIL STD 810C, Method 516.2			
Noise immunity	NEMA (ICS3-304)			
Weight	Products Number	Weight	Terminal Type	connector Type
	PZ1-8ND1-6TD1	110g		
	PZ1-8ND1-6TR1	150g		
	PZ1-8ND1-6TD1-A	190g		
	PZ1-8ND1-6TR1-A	220g		
	PZ2-16ND1-10TD1	180g		
	PZ2-16ND1-10TR1	250g		
	PZ2-16ND1-10TD1-A	275g		
	PZ2-16ND1-10TR1-A	325g		
	PZ3-16ND1-16TD1	240g		
	PZ3-T	150g		
	PZ3M	150g		
	PZ3E1	190g		
	PZ3E2			
	PZ3E3			

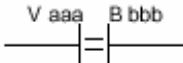
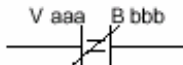
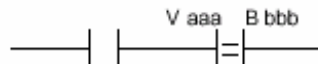
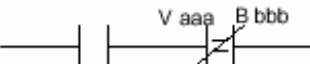


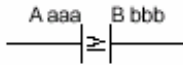
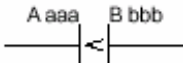
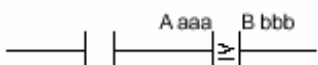
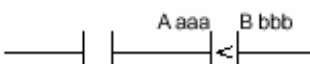
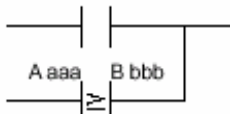
Function Specification


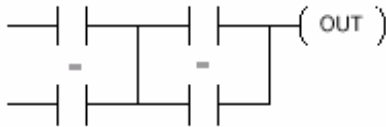
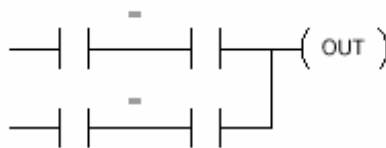
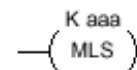
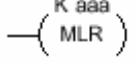
Items	PZ1	PZ2	PZ3
Ladder memory(words)	2048	2048	7680
V memory(words)	256	256	8320
Total IO	14	26	32
Input/ Output	8/6	16/10	16/16
IO Expansion	No	No	Yes
Typical scan	2.9ms under 200 boolean	1ms under 200 boolean	1.2ms under 200 boolean
Instructions and diagnostics			
Rll Ladder	Yes	Yes	Yes
Rll plus /Flowchart style	Yes	Yes	Yes
Scan	Variable	Variable	Variable
Instructions			
Control relays	256	256	1024
Timers	64	64	256
Counters	64	64	128
Immediate IO	In 128 / out 128	In 128 / out 128	In 512 / out 512
Subroutines	No	No	No
For / Next Loops	No	No	No
Up counter(5 kcps)	No	Yes	Yes
Up / Down counter(5 kcps)	No	Yes	Yes
Counter reset input	No	Yes	Yes
Pulse output (5 kpps)	No	Yes	Yes
Interrupt input(0.1ms)	No	Yes	Yes
Timer Interrupt(1ms)	No	Yes	Yes
Pulse catch input	No	Yes	Yes
Communications			
Serial port	1	1	2
Protocol	K-Sequence	K-Sequence	K-Sequence(port1) K-Sequence(port2) DirectNet(port2) Modbus RTU(port2)
Ascii out	No	No	Yes

Instructions

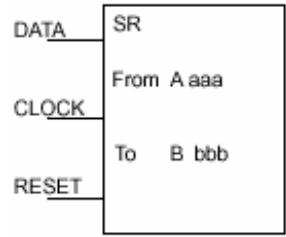
Boolean Instructions			
The Store instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the associated image register point or memory location.	Store (STR)		X,Y,C,T, CT,S,SP
The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the associated image register point or memory location.	Store Not (STRN)		X,Y,C,T, CT,S,SP
The And instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.	And (AND)		X,Y,C,T, CT,S,SP
The And Not instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.	And Not (ANDN)		X,Y,C,T, CT,S,SP
The Or instruction logically ors a normally open contact in parallel with another contact in a rung. The status of the contact will be the same state as the associated image register point or memory location.	Or (OR)		X,Y,C,T, CT,S,SP
The Or Not instruction logically ors a normally closed contact in parallel with another contact in a rung. The status of the contact will be opposite the state of the associated image register point or memory location.	Or Not (ORN)		T,CT,[K,V]
The Store Bit-of-Word instruction begins a new rung or an additional branch in a rung with a normally open contact. Status of the contact will be the same state as the bit referenced in the associated memory location.	Store Bit-of-Word (STRB)		T,CT,[K,V]
The Store Not instruction begins a new rung or an additional branch in a rung with a normally closed contact. Status of the contact will be opposite the state of the bit referenced in the associated memory location.	Store Not Bit-of-Word (STRNB)		T,CT,[K,V]
The And Bit-of-Word instruction logically ands a normally open contact in series with another contact in a rung. The status of the contact will be the same state as the bit referenced in the associated memory location.	And Bit-of-Word (ANDB)		T,CT,[K,V]

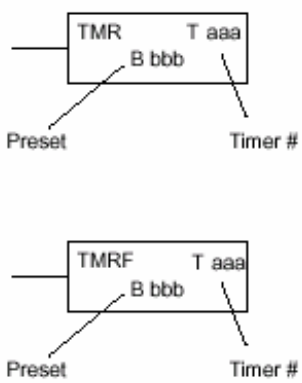
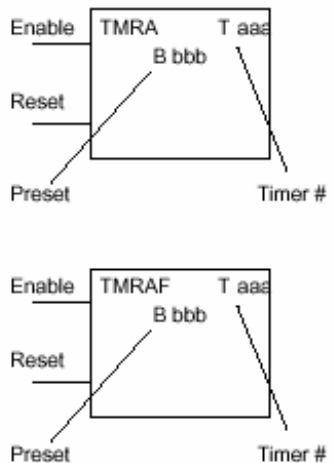
<p>The And Not Bit-of-Word instruction logically ands a normally closed contact in series with another contact in a rung. The status of the contact will be opposite the state of the bit referenced in the associated memory location.</p>	<p>And Not Bit-of-Word (ANDNB)</p>		<p>T,CT,[K,V]</p>
<p>The Or Bit-of-Word instruction logically ors a normally open contact in parallel with another contact in a rung. Status of the contact will be the same state as the bit referenced in the associated memory location.</p>	<p>Or Bit-of-Word (ORB)</p>		<p>T,CT,[K,V]</p>
<p>The Or Not Bit-of-Word instruction logically ors a normally closed contact in parallel with another contact in a rung. Status of the contact will be opposite the state of the bit referenced in the associated memory location.</p>	<p>Or Not Bit-of-Word (ORNB)</p>		<p>T,CT,[K,V]</p>

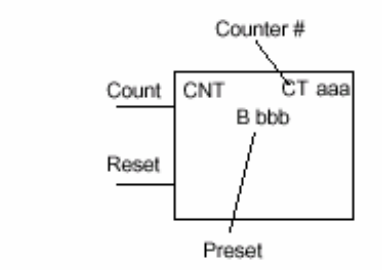
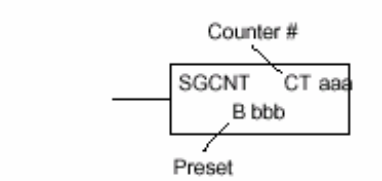
Comparative Boolean			
The Store If Equal instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when $V_{aaa} = B_{bbb}$.	Store If Equal (STRE)		V[K,V]
The Store If Not Equal instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when $V_{aaa} \neq B_{bbb}$.	Store If Not Equal (STRNE)		V[K,V]
The And If Equal instruction connects a normally open comparative contact in series with another contact. The contact will be on when $V_{aaa} = B_{bbb}$.	And If Equal (ANDE)		V[K,V]
The And If Not Equal instruction connects a normally closed comparative contact in series with another contact. The contact will be on when $V_{aaa} \neq B_{bbb}$.	And If Not Equal (ANDNE)		V[K,V]
The Or If Equal instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when $V_{aaa} = B_{bbb}$.	Or If Equal (ORE)		V[K,V]
The Or If Not Equal instruction connects a normally closed comparative contact in parallel with another contact. The contact will be on when $V_{aaa} \neq B_{bbb}$.	Or If Not Equal (ORNE)		V[K,V]
The Comparative Store instruction begins a new rung or additional branch in a rung with a normally open comparative contact. The contact will be on when $A_{aaa} \geq B_{bbb}$.	Store (STR)		V[K,V]
The Comparative Store Not instruction begins a new rung or additional branch in a rung with a normally closed comparative contact. The contact will be on when $A_{aaa} < B_{bbb}$.	Store Not (STRN)		V[K,V]
The Comparative And instruction connects a normally open comparative contact in series with another contact. The contact will be on when $A_{aaa} \geq B_{bbb}$.	And (AND)		V[K,V]
The Comparative And Not instruction connects a normally open comparative contact in series with another contact. The contact will be on when $A_{aaa} < B_{bbb}$.	And Not (ANDN)		V[K,V]
The Comparative Or instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when $A_{aaa} \geq B_{bbb}$.	Or (OR)		V[K,V]

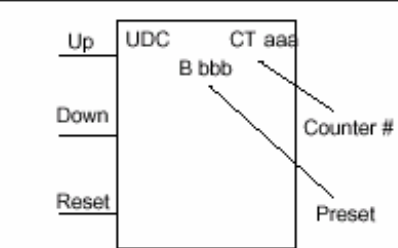
<p>The Comparative Or Not instruction connects a normally open comparative contact in parallel with another contact. The contact will be on when Aaaa < Bbbb.</p>	<p>Or Not (ORN)</p>		<p>V[K,V]</p>
<p>The And Store instruction logically ands two branches of a rung in series. Both branches must begin with the Store instruction.</p>	<p>And Store (AND STR)</p>		
<p>The Or Store instruction logically ors two branches of a rung in parallel. Both branches must begin with the Store instruction.</p>	<p>Or Store (OR STR)</p>		
<p>The Master Line Set instruction allows the program to control sections of ladder logic by forming a new power rail controlled by the main left power rail. The main left rail is always master line 0. When a MLS K1 instruction is used, a new power rail is created at level 1. Master Line Sets and Master Line Resets can be used to nest power rails up to seven levels deep. Note that unlike stages in RLLPLUS, the logic within the master control relays is still scanned and updated even though it will not function if the MLS is off.</p>	<p>Master Line Set (MLS)</p>		
<p>The Master Line Reset instruction marks the end of control for the corresponding MLS instruction. The MLR reference is one less than the corresponding MLS.</p>	<p>Master Line Reset (MLR)</p>		



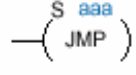
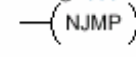


Out Instruction			
<p>The Out instruction reflects the status of the rung (on/off) and outputs the discrete (on/off) state to the specified image register point or memory location. Multiple Out instructions referencing the same discrete location should not be used since only the last Out instruction in the program will control the physical output point.</p>	Out (OUT)		X,Y,C
<p>The Or Out instruction has been designed to be used more than 1 rung of discrete logic to control a single output. Multiple Or Out instructions referencing the same output coil may be used, since <i>all</i> contacts controlling the output are ored together. If the status of <i>any</i> rung is on, the output will also be on.</p>	Or Out (OR OUT)		X,Y,C
<p>The Set instruction sets or turns on an image register point/memory location or a consecutive range of image register points/memory locations. Once the point/location is set it will remain on until it is reset using the Reset instruction. It is not necessary for the input controlling the Set instruction to remain on.</p>	Set (SET)		X,Y,C,S
<p>The Reset instruction resets or turns off an image register point/memory location or a range of image registers points/memory locations. Once the point/location is reset it is not necessary for the input to remain on.</p>	Reset (RST)		X,Y,C, T,CT,S
<p>The Positive Differential instruction is typically known as a one shot. When the input logic produces an off to on transition, the output will energize for one CPU scan.</p>	Positive Differential (PD)		X,Y,C



Shift Register			
<p>The Shift Register instruction shifts data through a predefined number of control relays. The control ranges in the shift register block must start at the beginning of an 8 bit boundary and end at the end of an 8 bit boundary.</p> <p>The Shift Register has three contacts.</p> <ul style="list-style-type: none"> - Data — determines the value (1 or 0) that will enter the register - Clock — shifts the bits one position on each low to high transition - Reset —resets the Shift Register to all zeros. 	<p>Shift Register (SR)</p>		<p>C(0-377)</p>
<p>With each off to on transition of the clock input, the bits which make up the shift register block are shifted by one bit position and the status of the data input is placed into the starting bit position in the shift register. The direction of the shift depends on the entry in the From and To fields. From C0 to C17 would define a block of sixteen bits to be shifted from left to right. From C17 to C0 would define a block of sixteen bits, to be shifted from right to left. The maximum size of the shift register block depends on the number of available control relays. The minimum block size is 8 control relays.</p>			

Timer			
<p>The Timer instruction is a 0.1 second single input timer that times to a maximum of 999.9 seconds. The Timer Fast instruction is a 0.01 second single input timer that times up to a maximum of 99.99 seconds. These timers will be enabled if the input logic is true (on) and will be reset to 0 if the input logic is false (off).</p> <p>Instruction Specifications Timer Reference (Taaa): Specifies the timer number. Preset Value (Bbbb): Constant value (K) or a V memory location. (Pointer (P) for DL240 or DL250 only.) Current Value: Timer current values are accessed by referencing the associated V or T memory location*. For example, the timer current value for T3 physically resides in V-memory location V3. Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. It will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.</p>	<p>Timer (TMR) and Timer Fast (TMRF)</p>		T[K,V]
<p>The Accumulating Timer is a 0.1 second two input timer that will time to a maximum of 99999.9. The Accumulating Fast Timer is a 0.01 second two input timer that will time to a maximum of 99.99. These timers have two inputs, an enable and a reset. The timer will start timing when the enable is on and stop timing when the enable is off without resetting the current value to 0. The reset will reset the timer when on and allow the timer to time when off.</p> <p>Instruction Specifications Timer Reference (Taaa): Specifies the timer number. Preset Value (Bbbb): Constant value (K) or a V memory location. (Pointer (P) for DL240 and DL250). Current Value: Timer current values are accessed by referencing the associated V or T memory location (See Note). For example, the timer current value for T3 resides in V-memory location V3. Discrete Status Bit: The discrete status bit is accessed by referencing the associated T memory location. It will be on if the current value is equal to or greater than the preset value. For example the discrete status bit for timer 2 would be T2.</p>	<p>Accumulating Timer (TMRA) and Accumulating Fast Timer (TMR AF)</p>	 <p>Caution: The TMRA uses two consecutive timer locations, since the preset can now be 8 digits, which requires two V-memory locations. For example, if TMRA T0 is used in the program, the next available timer would be T2. Or if T0 was a normal timer, and T1 was an accumulating timer, the next available timer would be T3.</p> <p>The timer discrete status bit and the current value are not specified in the timer instruction.</p>	T[K,V]

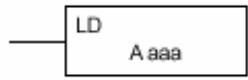
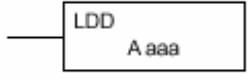
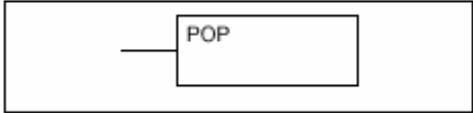
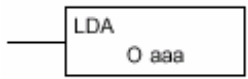
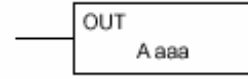
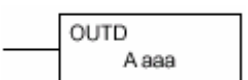
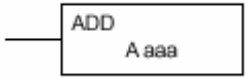
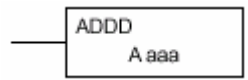
<p>The Counter is a two input counter that increments when the count input logic transitions from off to on. When the counter reset input is on the counter resets to 0. When the current value equals the preset value, the counter status bit comes on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.</p> <p>Instruction Specifications Counter Reference (CTaaa): Specifies the counter number. Preset Value (Bbbb): Constant value (K) or a V memory location. (Pointer (P) for DL240 and DL250). Current Values: Counter current values are accessed by referencing the associated V or CT memory locations*. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V memory location V1003. Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.</p>	<p>Counter (CNT)</p>	 <p>The counter discrete status bit and the current value are not specified in the counter instruction.</p>	<p>CT[K,V]</p>
<p>The Stage Counter is a single input counter that increments when the input logic transitions from off to on. This counter differs from other counters since it will hold its current value until reset using the RST instruction. The Stage Counter is designed for use in RLL^{PLUS} programs but can be used in relay ladder logic programs. When the current value equals the preset value, the counter status bit turns on and the counter continues to count up to a maximum count of 9999. The maximum value will be held until the counter is reset.</p> <p>Instruction Specifications Counter Reference (CTaaa): Specifies the counter number. Preset Value (Bbbb): Constant value (K) or a V memory location. (Pointer (P) for DL240 and DL250). Current Values: Counter current values are accessed by referencing the associated V or CT memory locations*. The V-memory location is the counter location + 1000. For example, the counter current value for CT3 resides in V memory location V1003. Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if the value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.</p>	<p>Stage Counter (SGCNT)</p>	 <p>The counter discrete status bit and the current value are not specified in the counter instruction.</p>	<p>CT[K,V]</p>

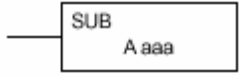
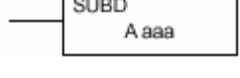
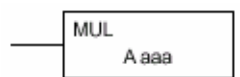
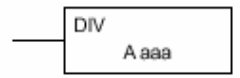
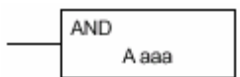
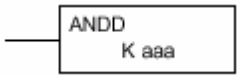
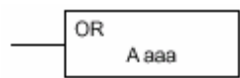
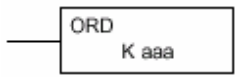
<p>This Up/Down Counter counts up on each off to on transition of the Up input and counts down on each off to on transition of the Down input. The counter is reset to 0 when the Reset input is on. The count range is 0–99999999. The count input not being used must be off in order for the active count input to function.</p> <p>Instruction Specification Counter Reference (CTaaa): Specifies the counter number. Preset Value (Bbbb): Constant value (K) or two consecutive V memory locations. Current Values: Current count is a double word value accessed by referencing the associated V or CT memory locations*. The V-memory location is the counter location + 1000. For example, the counter current value for CT5 resides in V memory location V1005 and V1006. Discrete Status Bit: The discrete status bit is accessed by referencing the associated CT memory location. It will be on if value is equal to or greater than the preset value. For example the discrete status bit for counter 2 would be CT2.</p>	<p>Up Down Counter (UDC)</p>	 <p>Caution : The UDC uses two V memory locations for the 8 digit current value. This means the UDC uses two consecutive counter locations. If UDC CT1 is used in the program, the next available counter is CT3.</p> <p>The counter discrete status bit and the current value are not specified in the counter instruction.</p>	
<p>The Reset instruction resets or turns off an image register point/memory location or a range of image registers points/memory locations. Once the point/location is reset it is not necessary for the input to remain on.</p>	<p>Reset (RST)</p>	<p>Optional memory range</p> <p>A aaa aaa</p> <p>— (RST)</p>	<p>X,Y,C, T,CT,S</p>

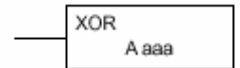
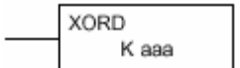
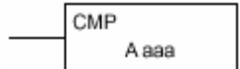
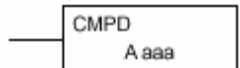
RLL plus Instruction			
<p>The Stage instructions are used to create structured RLL^{PLUS} programs. Stages are program segments which can be activated by transitional logic, a jump or a set stage that is executed from an active stage. Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed.</p>	<p>Stage (SG)</p>		<p>S</p>
<p>The Initial Stage instruction is normally used as the first segment of an RLL^{PLUS} program. Initial stages will be active when the CPU enters the run mode allowing for a starting point in the program. Initial Stages are also activated by transitional logic, a jump or a set stage executed from an active stage. Initial Stages are deactivated one scan after transitional logic, a jump, or a reset stage instruction is executed. Multiple Initial Stages are allowed in a program.</p>	<p>Initial Stage (ISG)</p>		<p>S</p>
<p>The Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which stage is specified in the instruction. The jump will occur when the input logic is true. The active stage that contains the Jump will be deactivated 1 scan after the Jump instruction is executed.</p>	<p>Jump (JMP)</p>		<p>S</p>
<p>The Not Jump instruction allows the program to transition from an active stage which contains the jump instruction to another which is specified in the instruction. The jump will occur when the input logic is off. The active stage that contains the Not Jump will be deactivated 1 scan after the Not Jump instruction is executed.</p>	<p>Not Jump (NJMP)</p>		<p>S</p>
<p>The Converge Stage instruction is used to group certain stages together by defining them as Converge Stages.</p> <p>When all of the Converge Stages within a group become active, the CVJMP instruction (and any additional logic in the final CV stage) will be executed. All preceding CV stages <i>must</i> be active before the final CV stage logic can be executed. All Converge Stages are deactivated one scan after the CVJMP instruction is executed.</p> <p>Additional logic instructions are only allowed following the last Converge Stage instruction and before the CVJMP instruction. Multiple CVJUMP instructions are allowed.</p> <p>Converge Stages must be programmed in the main body of the application program. This means they cannot be programmed in Subroutines or Interrupt Routines.</p>	<p>Converge Stage (CV) and Converge Jump (CVJMP)</p>	 	<p>S</p>





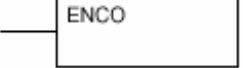

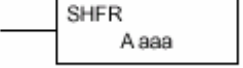
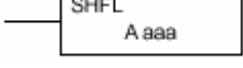
<p>The stage block instructions are used to activate a block of stages. The Block Call, Block, and Block End instructions must be used together.</p> <p>The BCALL instruction is used to activate a stage block. There are several things you need to know about the BCALL instruction.</p> <p>Uses CR Numbers — The BCALL appears as an output coil, but does not actually refer to a Stage number as you might think. Instead, the block is identified with a Control Relay (Caaa). This control relay cannot be used as an output anywhere else in the program.</p> <p>Must Remain Active — The BCALL instruction actually controls all the stages between the BLK and the BEND instructions even after the stages inside the block have started executing. The BCALL must remain active or all the stages in the block will automatically be turned off. <i>If either the BCALL instruction, or the stage that contains the BCALL instruction goes off, then the stages in the defined block will be turned off automatically.</i></p> <p>Activates First Block Stage — When the BCALL is executed it automatically activates the first stage following the BLK instructions.</p>		<p>Block Call (BCALL)</p>	<p>C</p>
<p>The Block instruction is a label which marks the beginning of a block of stages that can be activated as a group. A Stage instruction must immediately follow the Start Block instruction. Initial Stage instructions are not allowed in a block. The control relay (Caaa) specified in Block instruction must not be used as an output any where else in the program.</p>	<p>Block (BLK)</p>		
<p>The Block End instruction is a label used with the Block instruction. It marks the end of a block of stages. There is no operand with this instruction. Only one Block End is allowed per Block Call.</p>	<p>Block End (BEND)</p>		

Interrupt Instruction			
The Enable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to enable hardware or software interrupts. Once the coil has been energized interrupts will be enabled until the interrupt is disabled by the Disable Interrupt instruction.	Enable Interrupts (ENI)	—(ENI)	
The Disable Interrupt instruction is programmed in the main body of the application program (before the End instruction) to disable both hardware or software interrupts. Once the coil has been energized interrupts will be disabled until the interrupt is enabled by the Enable Interrupt instruction.	Disable Interrupts (DISI)	—(DISI)	
When an Interrupt Return is executed in the interrupt routine the CPU will return to the point in the main body of the program from which it was called. The Interrupt Return is programmed as the last instruction in an interrupt routine and is a stand alone instruction (no input contact on the rung).	Interrupt Return (IRT)	—(IRT)	
The Interrupt Return Conditional instruction is an optional instruction used with an input contact to implement a conditional return from the interrupt routine. The Interrupt Return is required to terminate the interrupt routine.	Interrupt Return Conditional (IRTC)	—(IRTC)	
The Stop instruction changes the operational mode of the CPU from Run to Program (Stop) mode. This instruction is typically used to stop PLC operation in a shutdown condition such as a I/O module failure.	Stop (STOP)	—(STOP)	
The No Operation is an empty (not programmed) memory location.	No Operation (NOP)	—(NOP)	
The End instruction marks the termination point of the normal program scan. An End instruction is required at the end of the main program body. If the End instruction is omitted an error will occur and the CPU will not enter the Run Mode. Data labels, subroutines and interrupt routines are placed after the End instruction. The End instruction is not conditional; therefore, no input contact is allowed.	End (END)	—(END)	

Accumulator / stock load and output data instructions			
<p>The Load instruction is a 16 bit instruction that loads the value (Aaaa), which is either a V memory location or a 4 digit constant, into the lower 16 bits of the accumulator. The upper 16 bits of the accumulator are set to 0.</p>	<p>Load (LD)</p>		V,P, K
<p>The Load Double instruction is a 32 bit instruction that loads the value (Aaaa), which is either two consecutive V memory locations or an 8 digit constant value, into the accumulator.</p>	<p>Load Double (LDD)</p>		V,P, K
<p>The Pop instruction moves the value from the first level of the accumulator stack (32 bits) to the accumulator and shifts each value in the stack up one level.</p> <p>In the example, when C0 is on, the value 4545 that was on top of the stack is moved into the accumulator using the Pop instruction. The value is output to V2000 using the Out instruction. The next Pop moves the value 3792 into the accumulator and outputs the value to V2001. The last Pop moves the value 7930 into the accumulator and outputs the value to V2002. Please note if the value in the stack were greater than 16 bits (4 digits) the Out Double instruction would be used and 2 V memory locations for each Out Double need to be allocated.</p>		<p>Pop (POP)</p>	
<p>The Load Address instruction is a 16 bit instruction. It converts any octal value or address to the HEX equivalent value and loads the HEX value into the accumulator. This instruction is useful when an address parameter is required since all addresses for the DL205 system are in octal.</p>	<p>Load Address (LDA)</p>		O
<p>The Out instruction is a 16 bit instruction that copies the value in the lower 16 bits of the accumulator to a specified V memory location (Aaaa).</p>	<p>Out (OUT)</p>		V,P
<p>The Out Double instruction is a 32 bit instruction that copies the value in the accumulator to two consecutive V memory locations at a specified starting location (Aaaa).</p>	<p>Out DOUBLE (OUTD)</p>		V,P
<p>Add is a 16 bit instruction that adds a BCD value in the accumulator with a BCD value in a V memory location (Aaaa). The result resides in the accumulator.</p>	<p>Add (ADD)</p>		V
<p>Add Double is a 32 bit instruction that adds the BCD value in the accumulator with a BCD value (Aaaa), which is either two consecutive V memory locations or an 8-digit (max.) BCD constant. The result resides in the accumulator.</p>	<p>Add Double (ADDD)</p>		V,K

<p>Subtract is a 16 bit instruction that subtracts the BCD value (Aaaa) in a V memory location from the BCD value in the lower 16 bits of the accumulator. The result resides in the accumulator.</p>	<p>Subtract (SUB)</p>		<p>V</p>
<p>Subtract Double is a 32 bit instruction that subtracts the BCD value (Aaaa), which is either two consecutive V memory locations or an 8-digit (max.) constant, from the BCD value in the accumulator. The result resides in the accumulator.</p>	<p>Subtract Double (SUBD)</p>		<p>V,K</p>
<p>Multiply is a 16 bit instruction that multiplies the BCD value (Aaaa), which is either a V memory location or a 4-digit (max.) constant, by the BCD value in the lower 16 bits of the accumulator. The result can be up to 8 digits and resides in the accumulator.</p>	<p>Multiply (MUL)</p>		<p>V,K</p>
<p>Divide is a 16 bit instruction that divides the BCD value in the accumulator by a BCD value (Aaaa), which is either a V memory location or a 4-digit (max.) constant. The first part of the quotient resides in the accumulator and the remainder resides in the first stack location.</p>	<p>Divide (DIV)</p>		<p>V,K</p>
<p>The And instruction is a 16 bit instruction that logically ands the value in the lower 16 bits of the accumulator with a specified V memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the And is zero.</p>	<p>And (AND)</p>		<p>V</p>
<p>The And Double is a 32 bit instruction that logically ands the value in the accumulator with an 8 digit (max.) constant value (Aaaa). The result resides in the accumulator. Discrete status flags indicate if the result of the And Double is zero or a negative number (the most significant bit is on).</p>	<p>And Double (ANDD)</p>		<p>K</p>
<p>The Or instruction is a 16 bit instruction that logically ors the value in the lower 16 bits of the accumulator with a specified V memory location (Aaaa). The result resides in the accumulator. The discrete status flag indicates if the result of the Or is zero.</p>	<p>Or (OR)</p>		<p>V</p>
<p>The Or Double is a 32 bit instruction that ors the value in the accumulator with the value (Aaaa), or an 8 digit (max.) constant value. The result resides in the accumulator. Discrete status flags indicate if the result of the Or Double is zero or a negative number (the most significant bit is on).</p>	<p>Or Double (ORD)</p>		<p>K</p>

<p>The Exclusive Or instruction is a 16 bit instruction that performs an exclusive or of the value in the lower 16 bits of the accumulator and a specified V memory location (Aaaa). The result resides in the in the accumulator. The discrete status flag indicates if the result of the XOR is zero.</p>	<p>Exclusive Or (XOR)</p>		<p>V</p>
<p>The Exclusive OR Double is a 32 bit instruction that performs an exclusive or of the value in the accumulator and the value (Aaaa), which is a 8 digit (max.) constant. The result resides in the accumulator. Discrete status flags indicate if the result of the Exclusive Or Double is zero or a negative number (the most significant bit is on).</p>	<p>Exclusive Or Double (XORD)</p>		<p>K</p>
<p>The compare instruction is a 16 bit instruction that compares the value in the lower 16 bits of the accumulator with the value in a specified V memory location (Aaaa). The corresponding status flag will be turned on indicating the result of the comparison.</p>	<p>Compare (CMP)</p>		<p>V</p>
<p>The Compare Double instruction is a 32-bit instruction that compares the value in the accumulator with the value (Aaaa), which is either two consecutive V memory locations or an 8-digit (max.) constant. The corresponding status flag will be turned on indicating the result of the comparison.</p>	<p>Compare Double (CMPD)</p>		<p>V,K</p>

Number Conversion Instruction (Accumulator)			
The Invert instruction inverts or takes the one's complement of the 32 bit value in the accumulator. The result resides in the accumulator.	Invert (INV)		
The Binary instruction converts a BCD value in the accumulator to the equivalent binary value. The result resides in the accumulator.	Binary (BIN)		
The Binary Coded Decimal instruction converts a binary value in the accumulator to the equivalent BCD value. The result resides in the accumulator.	Binary Coded Decimal (BCD)		
The Ten's Complement instruction takes the 10's complement (BCD) of the 8 digit accumulator. The result resides in the accumulator. The calculation for this instruction is : $\begin{array}{r} 10000000 \\ - \text{accumulator value} \\ \hline \text{10's complement value} \end{array}$	Ten's Complement (BCDCPL)		
The Encode instruction encodes the bit position in the accumulator having a value of 1, and returns the appropriate binary representation. If the most significant bit is set to 1 (Bit 31), the Encode instruction would place the value HEX 1F (decimal 31) in the accumulator. If the value to be encoded is 0000 or 0001, the instruction will place a zero in the accumulator. If the value to be encoded has more than one bit position set to a "1", the least significant "1" will be encoded and SP53 will be set on.	Encode (ENCO)		
The Decode instruction decodes a 5 bit binary value of 0–31 (0–1F HEX) in the accumulator by setting the appropriate bit position to a 1. If the accumulator contains the value F (HEX), bit 15 will be set in the accumulator. If the value to be decoded is greater than 31, the number is divided by 32 until the value is less than 32 and then the value is decoded.	Decode (DECO)		
Shift Right is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the right. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.	Shift Right (SHFR)		
Shift Left is a 32 bit instruction that shifts the bits in the accumulator a specified number (Aaaa) of places to the left. The vacant positions are filled with zeros and the bits shifted out of the accumulator are lost.	Shift Left (SHFL)		

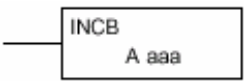
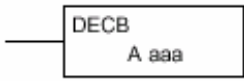
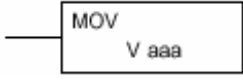
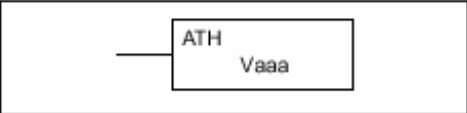
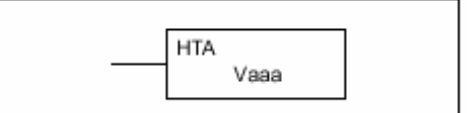
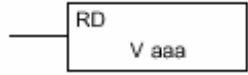
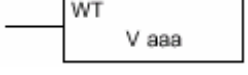
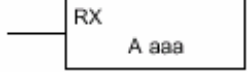
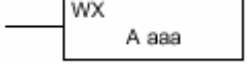
Register Increment Decrement			
The Increment Binary instruction increments a binary value in a specified V memory location by "1" each time the instruction is executed.	Increment Binary (INCB)		V
The Decrement Binary instruction decrements a binary value in a specified V memory location by "1" each time the instruction is executed.	Decrement Binary (DECB)		V

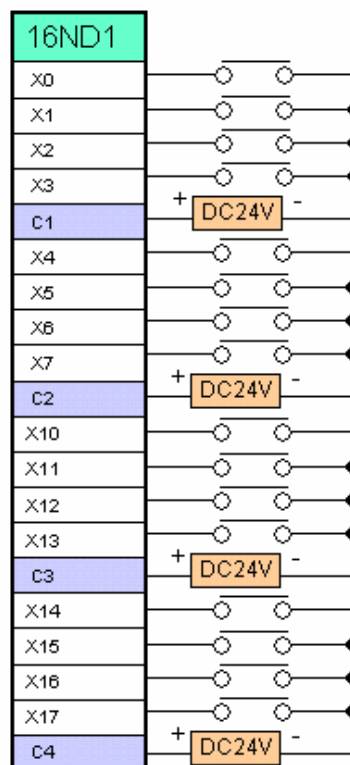
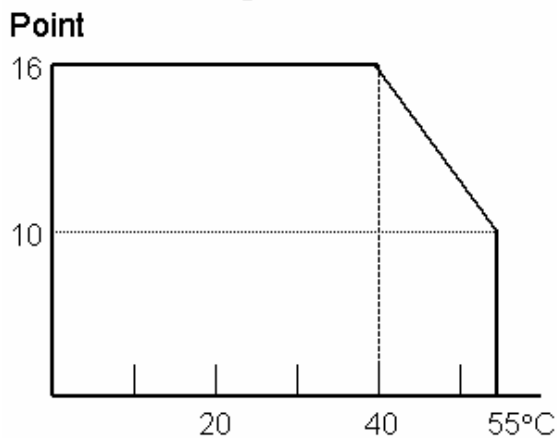
Table Instruction			
The Move instruction moves the values from a V memory table to another V memory table the same length. The function parameters are loaded into the first level of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move function.	Move (MOV)		V
The ASCII TO HEX instruction converts a table of ASCII values to a specified table of HEX values. ASCII values are two digits and their HEX equivalents are one digit. This means an ASCII table of four V memory locations would only require two V memory locations for the equivalent HEX table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program an ASCII to HEX table function. The example on the following page shows a program for the ASCII to HEX table function.		ASCII to HEX (ATH)	V
The HEX to ASCII instruction converts a table of HEX values to a specified table of ASCII values. HEX values are one digit and their ASCII equivalents are two digits. This means a HEX table of two V memory locations would require four V memory locations for the equivalent ASCII table. The function parameters are loaded into the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program a HEX to ASCII table function. The example on the following page shows a program for the HEX to ASCII table function.		HEX to ASCII (HTA)	V

Data label Instruction			
<p>The Data Label instruction marks the beginning of an ASCII / numeric data area. DLBLs are programmed after the End statement. A maximum of 64 (DL240 and DL250) or 32 (DL230) DLBL instructions can be used in a program. Multiple NCONs and ACONs can be used in a DLBL area.</p>	Data Label (DLBL)	<div style="border: 1px solid black; padding: 5px; width: fit-content;">DLBL K aaa</div>	K
<p>The Numerical Constant instruction is used with the DLBL instruction to store the HEX ASCII equivalent of numerical data for use with other instructions. Two digits can be stored in an NCON instruction.</p>	ASCII Constant (ACON)	<div style="border: 1px solid black; padding: 5px; width: fit-content;">NCON K aaa</div>	K
<p>The ASCII Constant instruction is used with the DLBL instruction to store ASCII text for use with other instructions. Two ASCII characters can be stored in an ACON instruction. If only one character is stored in a ACON a leading space will be printed in the Fault message.</p>	ASCII Constant (ACON)	<div style="border: 1px solid black; padding: 5px; width: fit-content;">ACON A aaa</div>	A
<p>The Move Memory Cartridge instruction is used to copy data between V memory and program ladder memory. The Load Label instruction is <i>only</i> used with the MOVMC instruction when copying data <i>from</i> program ladder memory <i>to</i> V memory.</p> <p>To copy data between V memory and program ladder memory, the function parameters are loaded into the first two levels of the accumulator stack and the accumulator by two additional instructions. Listed below are the steps necessary to program the Move Memory Cartridge and Load Label functions.</p>	Move Memory Cartridge / Load Label (MOVMC) (LDLBL)	<div style="display: flex; flex-direction: column; gap: 10px;"> <div style="border: 1px solid black; padding: 5px; width: fit-content;">LDLBL K aaa</div> <div style="border: 1px solid black; padding: 5px; width: fit-content;">MOVMC V aaa</div> </div>	K V

Module communication Instruction			
<p>The Read from Intelligent Module instruction reads a block of data (1–128 bytes maximum) from an intelligent I/O module into the CPU's V memory. It loads the function parameters into the first and second level of the accumulator stack, and the accumulator by three additional instructions.</p>	<p>Read from Intelligent Module (RD)</p>		<p>V</p>
<p>The Write to Intelligent Module instruction writes a block of data (1–128 bytes maximum) to an intelligent I/O module from a block of V memory in the CPU. The function parameters are loaded into the first and second level of the accumulator stack, and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Intelligent module function.</p>	<p>Write to Intelligent Module (WT)</p>		<p>V</p>
<p>The Read from Network instruction is used by the master device on a network to read a block of data from another CPU. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Read from Intelligent module function.</p>	<p>Read from Network (RX)</p>		<p>V</p>
<p>The Write to Network instruction is used to write a block of data from the master device to a slave device on the same network. The function parameters are loaded into the first and second level of the accumulator stack and the accumulator by three additional instructions. Listed below are the steps necessary to program the Write to Network function.</p>	<p>Write to Network (WX)</p>		<p>V</p>

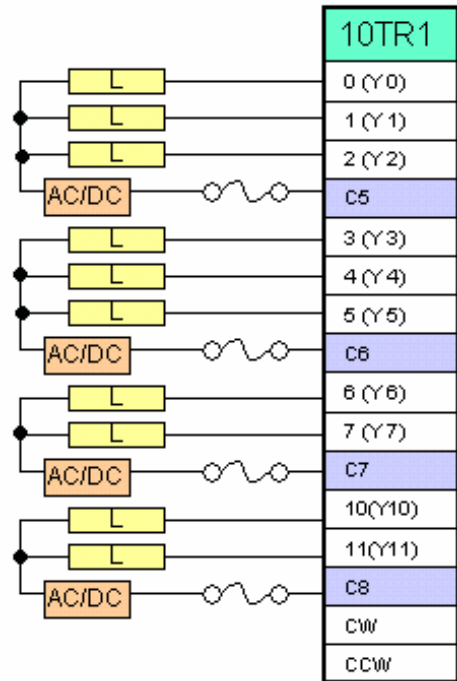
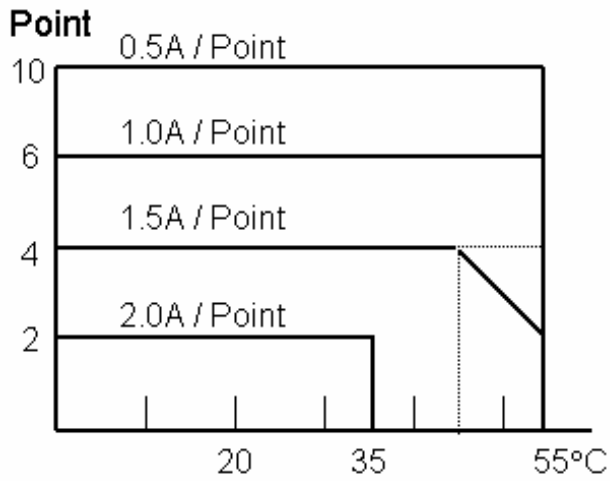
16ND1 DC Input (PZ2-16ND1-10TD1 / PZ2-16ND1-10TR1 / PZ3-T)		
Input per module	16 point	
Input type	X0 – X3 (sink) Choice from as follows 1) High speed counter 2) Pulse catch 3) Interrupt 4) Normal Input X4 – X7 X10-X17 (sink / source)	
Commons per module	4(Isolated)	
Input voltage range	2- 28VDC	
Typical Input voltage	X0-X3	12/24VDC
	X4 – X7 X10-X17	24VDC
ON voltage level	X0, 1	8.0VDC minimum
	X2, 3	9.5VDC minimum
	X4 – X7 X10-X17	18.5VDC minimum
OFF voltage level	X0, 1	1.0VDC maximum
	X2, 3	4.5VDC maximum
	X4 – X7 X10-X17	10.0VDC maximum
Input impedance	5.0 K	
Input current	X0, 1	10mA @ (12/24VDC)
	X2, 3	2.4mA @ 12VDC 4.8mA @ 24VDC
	X4 – X7 X10-X17	4.8mA @ 24VDC
Minimum ON current	X0, 1	8.0mA
	X2, 3	2.0mA
	X4 – X7 X10-X17	3.9mA
OFF to ON response	5ms (Default) *1	
ON to OFF response	5ms (Default) *1	
*1 Can be change 5 to 20 ms (1ms units) each 8point (X0 – X7 and X10-X17)		
Terminal type	Removable	
Status Indicator	On board	

Derating Chart

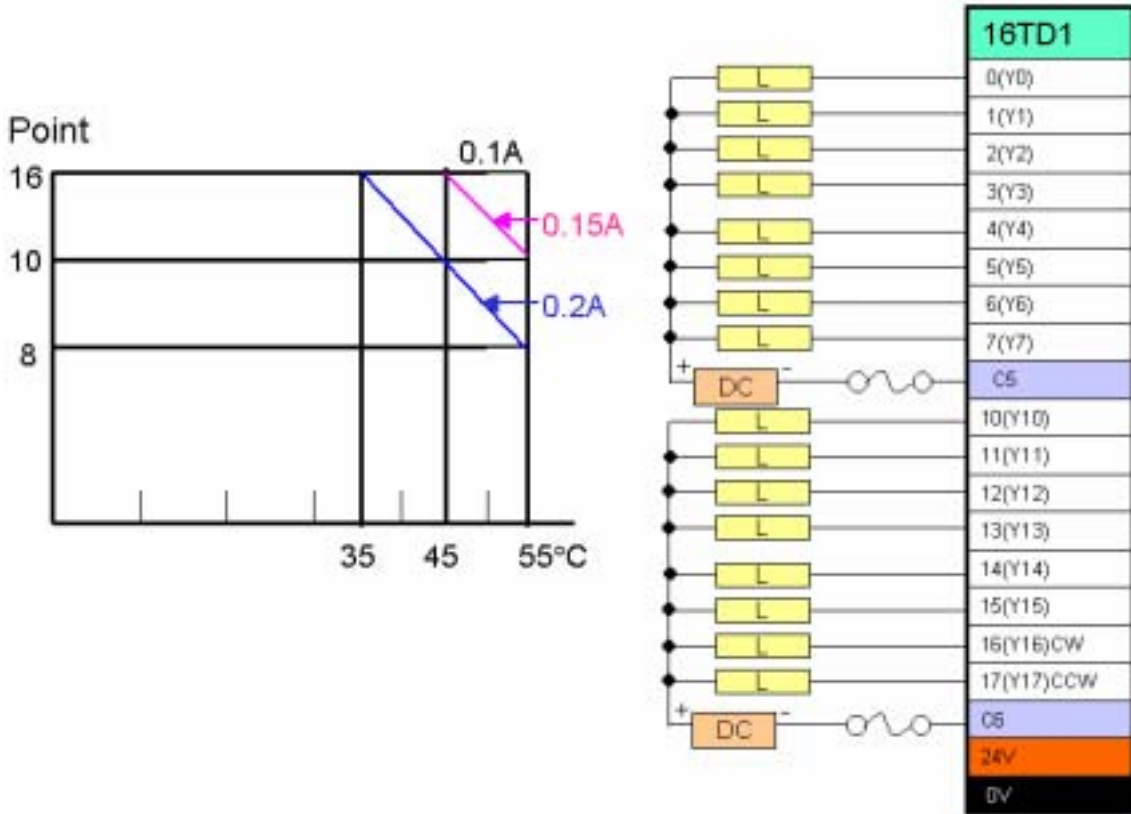


10TR1 Relay Output (PZ2-16ND1-10TR1)	
Output per module	10 point
Commons per module	4(Isolated)
Operating voltage	5- 24VDC 5- 240VAC
Output type	Relay, from A (SPAT)
Peak voltage	240V
AC frequency	47 to 60 Hz
Max current (resistive)	2A/ point 5A/ common
Max leakage current	<0.1mA @ 240V
Minimum load	10mA @ 4.5VDC
OFF to ON response	< 20ms
ON to OFF response	< 20ms
Terminal type	Removable
Status Indicator	On board
Fuses	No

Derating Chart



16TD1 DC Output (PZ3-16ND1-16TD1 / PZ3-T / PZ3M)	
Output per module	16 point
Commons per module	2(Isolated)
Operating voltage	5 / 12 / 24VDC
Output type	NPN open collector
Peak voltage	30V
AC frequency	N / A
Max Load current	0.3A / point
Max leakage current	<0.1mA @ 30VDC
Minimum load	10mA @ 4.5VDC
OFF to ON response	< 0.5ms
ON to OFF response	< 0.5ms
Terminal type	Removable
Status Indicator	On board
Fuses	No



Expansion board

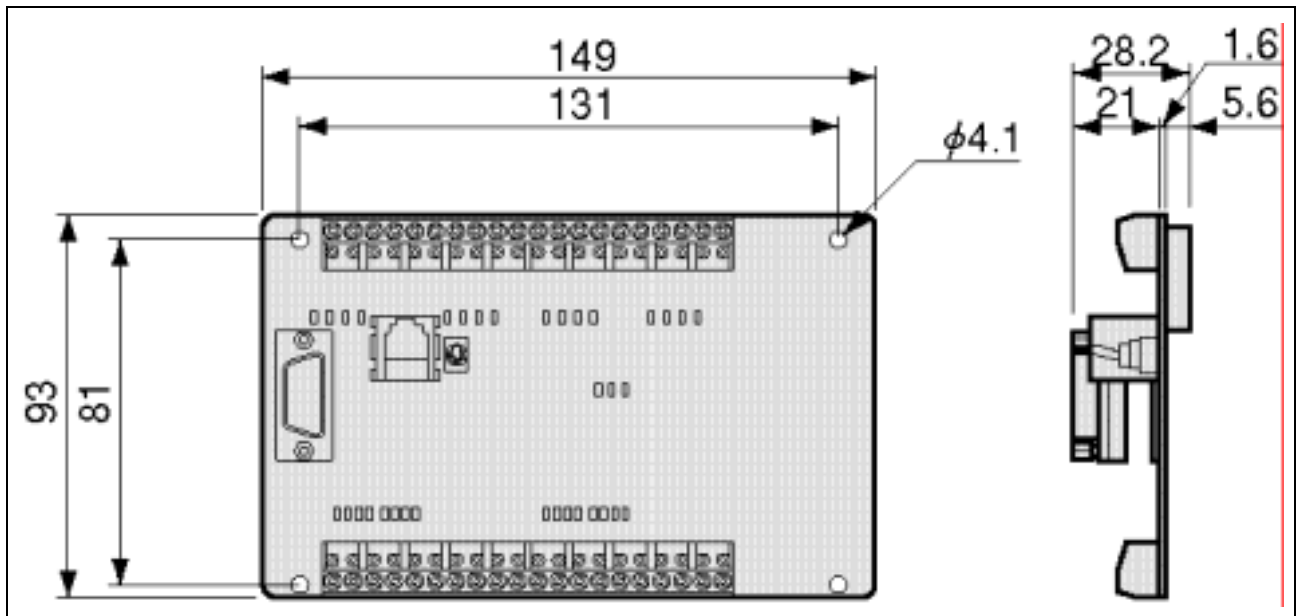
Expansion board for PZ3

Products Number	Input point	Output point	Analog
PZ3E1	DC 8	DC 8	AD 2ch
PZ3E2	DC 32	DC 32	None
PZ3E3	DC 40	None	None

PZ3 can be use IO combination

Ex type	PZ3 CPU	Ex1	Ex2	Ex3	Total IO point
Type1	PZ3 CPU	PZ3E1			In24/Out24 AD 2ch
Type2	PZ3 CPU	PZ3E2			In48/Out48
Type3	PZ3 CPU	PZ3E2	PZ3E2		In80/Out80
Type4	PZ3 CPU	PZ3E2	PZ3E3		In88/Out48
Type5	PZ3 CPU	PZ3E2	PZ3E2	PZ3E3	In120/Out80

*Can not use PZ3 and PZ3E3 of combination



PZ3-T

Expansion board for PZ3 DC Input (PZ3E1)	
Input per module	8 point
Input type	Sink / source
Commons per module	1(Isolated)
Input voltage range	19.2- 28.8VDC
Typical Input voltage	24VDC
ON voltage level	18.5VDC minimum
OFF voltage level	10.0VDC maximum
Input impedance	5.0 K
Input current	4.8mA @ 24VDC
Minimum ON current	3.9mA
Maximum OFF current	2.0mA
OFF to ON response	5-10ms
ON to OFF response	5-10ms
Terminal type	Removable
Status Indicator	On board

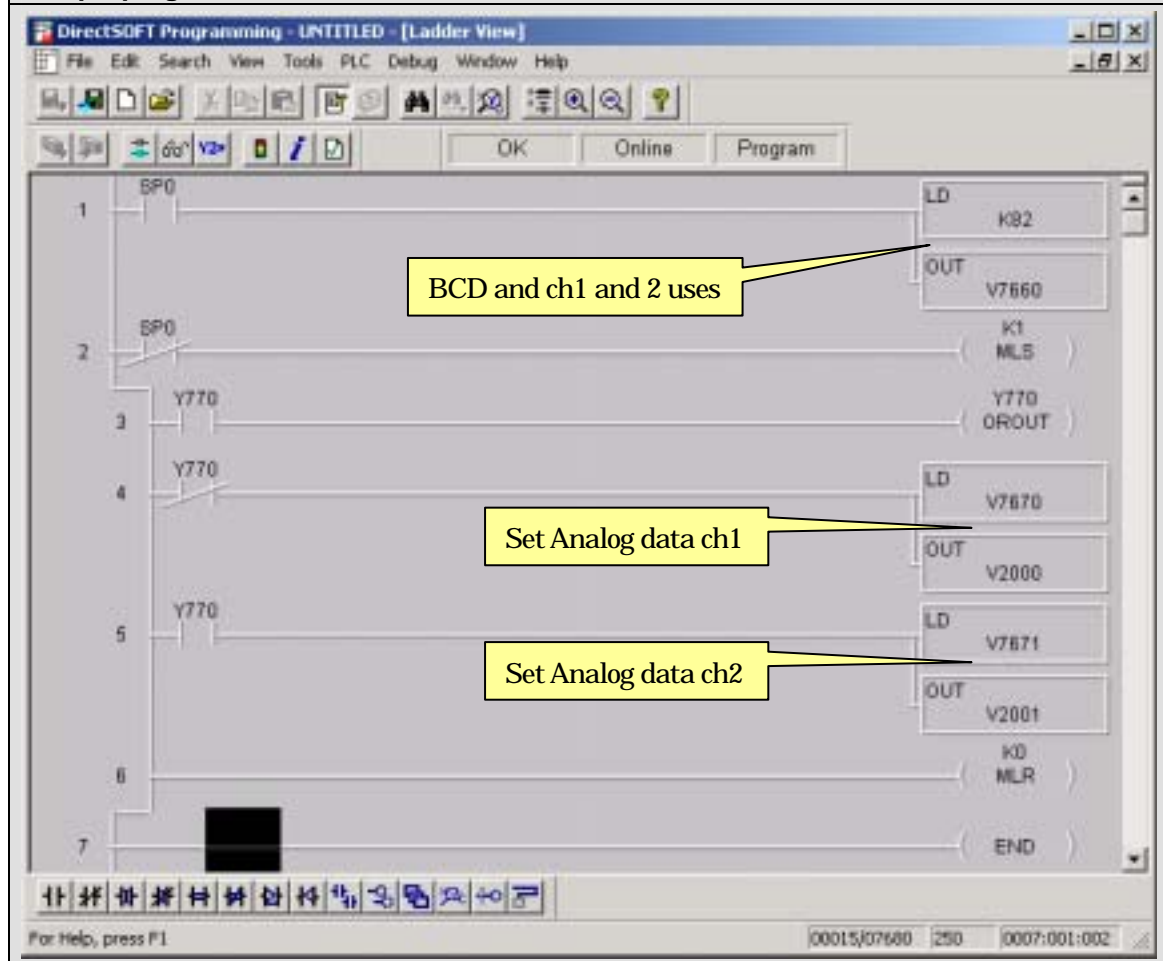
Expansion board for PZ3 DC Output (PZ3E1)	
Output per module	8 point
Commons per module	1(Isolated)
Operating voltage	5 / 12 / 24VDC
Output type	NPN open collector
Peak voltage	30V
AC frequency	N / A
Max Load current	0.3A / point
Max leakage current	<0.1mA @ 30VDC
Minimum load	10mA @ 4.5VDC
OFF to ON response	< 0.5ms
ON to OFF response	< 0.5ms
Terminal type	Removable
Status Indicator	On board
Fuses	No

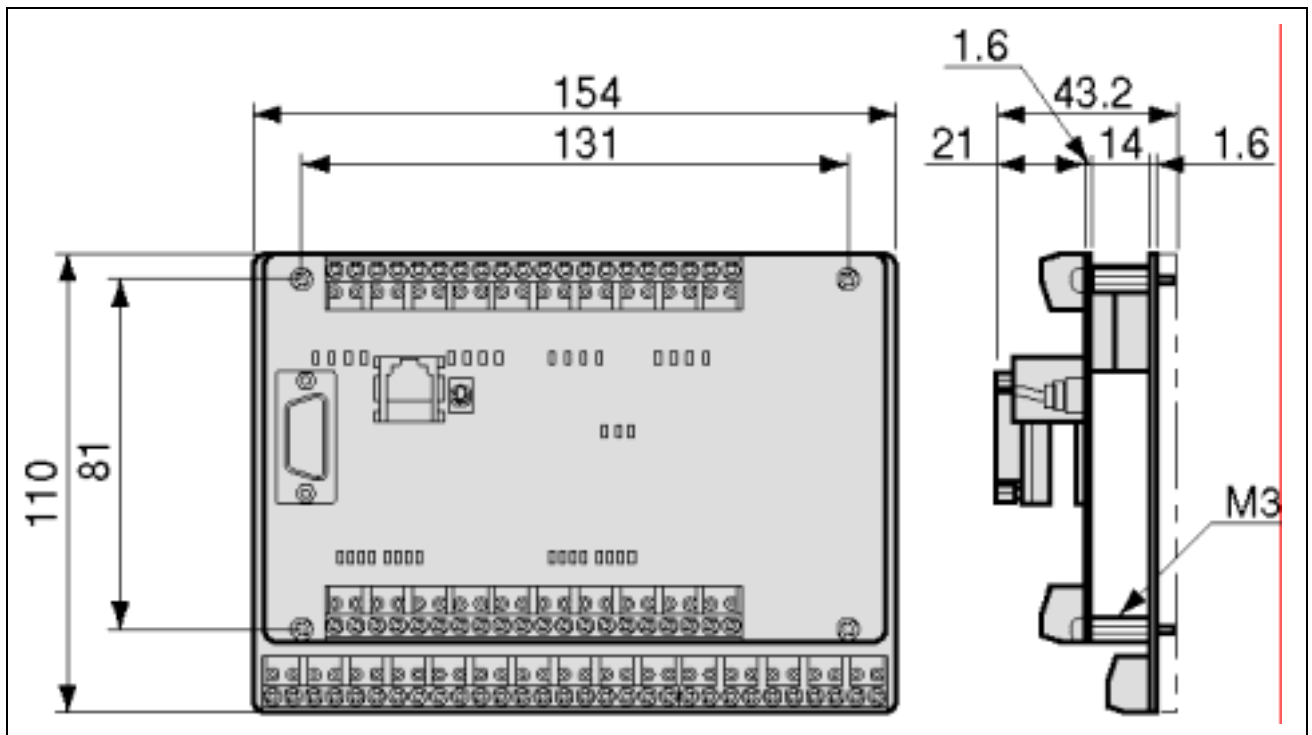
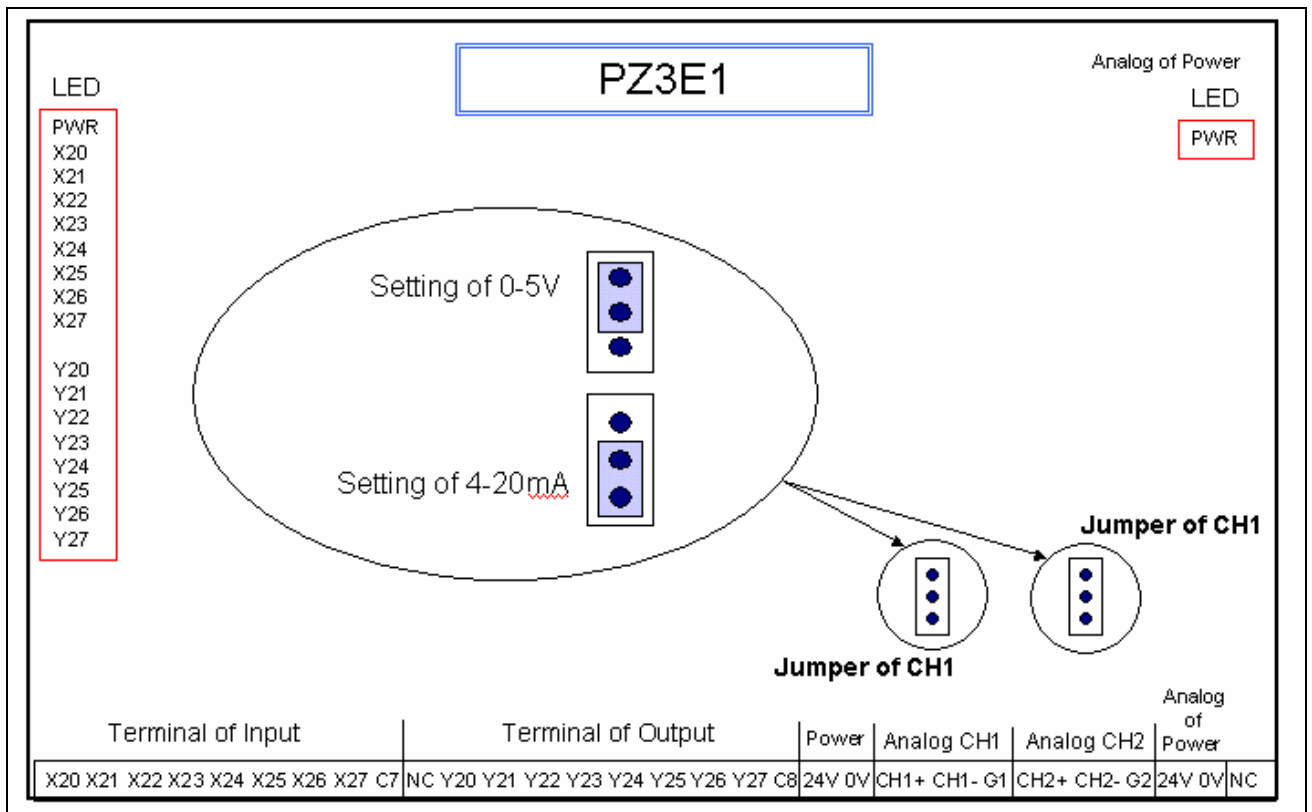
Expansion board for PZ3 Analog Input (PZ3E1)	
Number of Channels	2ch
Input Ranges	4-20mA current 0-5V Voltage
Input Impedance	4-20mA / 312 +/-2% 0-5V / >20M
Resolution	12bit (1 in 4096)
Linearity Error	1%
Total Error	1%
Convert type	Successive approximation
Conversion time (PLC update rate)	1channel per scan minimum
External Power Supply	50mA Maximum 21.6VDC to 26.4VDC
Terminal type	Removable
Status Indicator	None

Configuration V Memory for Analog Input

V Memory	Bit	Description
V7660	Bit 0	On = use ch1
	Bit 1	On = use ch1 and 2
	Bit 2-6	Always Zero
	Bit 7	Off = Set of analog data at BCD On = Set of analog data at BIN
	Bit 8-15	Not use
V7670	Bit 0-15	Analog data of ch1
V7671	Bit 0-15	Analog data of ch2

Sample program





PZ3-T and PZ3E1

Expansion board for PZ3 DC Input (PZ3E2)	
Input per module	32 point
Input type	Sink / source
Commons per module	2(Isolated)
Input voltage range	19.2- 28.8VDC
Typical Input voltage	24VDC
ON voltage level	18.5VDC minimum
OFF voltage level	10.0VDC maximum
Input impedance	5.0 K
Input current	4.8mA @ 24VDC
Minimum ON current	3.9mA
Maximum OFF current	2.0mA
OFF to ON response	4-6ms
ON to OFF response	4-6ms
Terminal type	Connector (MIL-C-83503) 40pin
Status Indicator	On board (16 point by change SW)

Expansion board for PZ3 DC Output (PZ3E2)	
Output per module	32 point
Commons per module	2(Isolated)
Operating voltage	5 / 12 / 24VDC
Output type	NPN open collector
Peak voltage	30V
AC frequency	N / A
Max Load current	0.3A / point
Max leakage current	<0.1mA @ 30VDC
Minimum load	10mA @ 4.5VDC
OFF to ON response	< 0.5ms
ON to OFF response	< 0.5ms
Terminal type	Connector (MIL-C-83503) 40pin
Status Indicator	On board (16 point by change SW)
Fuses	No

Expansion board for PZ3 DC Input (PZ3E3)	
Input per module	40 point
Input type	Sink / source
Commons per module	3(Isolated) 16point*2 and 8point*1
Input voltage range	19.2- 28.8VDC
Typical Input voltage	24VDC
ON voltage level	18.5VDC minimum
OFF voltage level	10.0VDC maximum
Input impedance	5.0 K
Input current	4.8mA @ 24VDC
Minimum ON current	3.7mA
Maximum OFF current	2.0mA
OFF to ON response	4-6ms
ON to OFF response	4-6ms
Terminal type	Connector (MIL-C-83503) 40pin / 20pin
Status Indicator	On board (24 /16 point by change SW)