



Greetings.

This file contains the answers to our **Email PLC Quiz #121**. This edition is a **Beginner Level** quiz which focuses on how programs which work perfectly in normal day-to-day operations can give surprising results when the PLC system experiences various "Go-To-Run" conditions.

If you'd like to discuss the information contained in any of our quizzes, please feel free to contact us. We'll be glad to answer any questions that you might have.

The following equipment was used during the research and development of this edition of the Email PLC Quiz. Other configurations might possibly give different results.

PLC-5/20E Processor; 1785-L20E/E; Series E; Rev H01; F/W Revision D.2
SLC-5/05 Processor; 1747-L552; Series A; OS501
MicroLogix-1000 Processor; 1761-L16AWA; Series A; FRN 2.0
MicroLogix-1000 Processor; 1761-L10BXB; Series F; FRN 1.1
MicroLogix-1500 Processor; 1764-LSP; Series C
MicroLogix-1500 Base Unit; 1764-24AWA; Series A; Rev A
ControlLogix-5555 Processor; 1756-L55/A 1756-M14/A; Revision 15.4

1771-IAD; 1746-IA16 - Series C; 1746-OB16 - Series D; 1746-OA16 - Series D;
1756-IA16/A - Rev 2.5; 1756-OA16/A - Rev 2.3; 1756-IB16I/A - Rev 2.1;
1756-OB32/A - Rev 2.4;

RSLogix5 Software - Version 6.00.00; RSLogix500 Software - Version 6.00.00;
RSLogix5000 Software - Version 15.00.00; RSLinx Lite Software - Version 2.50.00.20 (CPR 7);
Microsoft Windows 2000 Software - Version 5.00.2195;

A number of readers have responded to previous editions of our Email PLC Quizzes with requests for any books that we might have available covering this type of material. At this point in time, we are providing training mainly through our five-day PLC Boot Camp classes. So far we haven't found a satisfactory way to effectively present the same amount of detail in any format other than through face-to-face hands-on training. So no, we're sorry that we don't have any books or other types of training materials to offer - but we thank you very much for asking.

Beginner Level Quiz #121 - Answers to the Questions

Question 1 - Answer C
Question 2 - Answer C
Question 3 - Answer A
Question 4 - Answer C
Question 5 - Answer B
Question 6 - Answer F
Question 7 - Answer D
Question 8 - Answer B
Question 9 - Answer A
Question 10 - Answer C
Question 11 - Answer C
Question 12 - Answer D
Question 13 - Answer B
Question 14 - Answer D
Question 15 - Answer D
Question 16 - Answer B
Question 17 - Answer A

Beginner Level Quiz #121 - Points to Ponder

(1) In Question 1, going from Program Mode to Run Mode causes PUMP_2 to improperly start up and run for 5 seconds - even if SWITCH_A in the field is left OFF. Note that PUMP_1 does NOT run under the same test conditions.

(2) In Question 2, a full power cycle causes PUMP_2 to improperly start up and run for 5 seconds - even if SWITCH_A in the field is left OFF. Note that PUMP_1 does NOT run under the same test conditions.

(3) In Question 3, a brief power flicker does NOT cause PUMP_2 to run for 5 seconds - which is different from its operation after the full power cycle test used in Question 2.

(4) In Question 4, a brief power flicker DOES cause PUMP_2 to improperly start up and run for 5 seconds - which is different from the same flicker test made in Question 3. The different results are caused by using a PLC-5 processor instead of an MLX-1000.

(5) Notice that PUMP_1 never improperly started up and ran after any of the tests in Questions 1 through 4 - although PUMP_2 frequently DID. The different results are caused by the TOF and TON instructions. Remember that BOTH programming methods acted identically in normal day-to-day operations. The improper operations of PUMP_2 only surfaced at "Go-To-Run" time.

(6) In Question 5, notice that a "constantly TRUE" condition will NOT cause a CTU to count - which is different from the operation of the "constantly TRUE" ADD instruction. Notice that the ADD instruction can cause a fault if allowed to go too high. Many programmers use an unconditional OTU (Unlatch) rung for S:5/0 at the end of Ladder File #2 to prevent shutting down the system. The wisdom of this practice is often debated.

(7) In Question 6, notice that the ADD instruction will NOT cause a fault if allowed to run continuously - which is different from the operation of the same ADD instruction in Question 5. The different results are caused by using a PLC-5 processor instead of an MLX-1000.

(8) In Question 7, notice that the CTU will NOT count the "First Pass" events as intended - but the ADD instruction WILL count each of the separate "First Pass" events.

(9) In Question 8, notice that the ADD instruction will count ONLY the very first "Go-To-Run" event - which is different from the operation of the similar ADD rung in Question 7. Unless the status of bit B3/8 is somehow changed from a 1 to a 0, the OSR in Rung 0001 will never "fire" again.

(10) In Question 9, notice that leaving SWITCH_A in the ON position will NOT "fire" the OSR B3/10 when going from Program to Run Mode. Although it appears similar, this test is different from the "constantly TRUE" conditions of Rung 0001 in Question 8. When the "Go-To-Run" test of Question 9 began, the OSR bit B3/10 already had a "fired" status of 1 - since the processor had previously been placed in the Run Mode.

(11) In Question 10, notice that having SWITCH_A in the OFF position at the beginning of the test WILL "fire" the OSR B3/10 when going back to the Run Mode. This is different from the results of Question 9 where the switch was already ON at the beginning of the test.

(12) In Question 11, notice that a full power cycle DOES cause the ADD instruction to increment - but the CTU instruction does NOT increment.

(13) In Question 12, notice that a brief power flicker DOES cause the CTU to increment - which is different from the results of the full power cycle in Question 11 where the same CTU did NOT increment. This illustrates how in some cases a brief flicker can actually cause greater disruptions than a full power cycle.

(14) Notice that the ADD instruction for N7:10 did NOT increment in Question 9 - but it DID increment in Question 11. The different results were caused by changing the mode from Run to Program and back to Run in Question 9 - as compared to a full power cycle in Question 11. This illustrates that using a simple "mode change" is not always a conclusive way to test a program against all possible "Go-To-Run" situations.

(15) In Question 13, notice that the "home made one-shot" arrangement DID trigger its ADD to increment N7:11 - which is different from the results of the OSR which did NOT produce a trigger for N7:12. Remember that BOTH programming methods acted identically in normal day-to-day operations. The difference in operations only surfaced at "Go-To-Run" time.

(16) Question 13 is another example of how two programming methods can each give perfectly reliable results when tested in normal day-to-day operations - and yet give different results when put through certain "Go-To-Run" situations. Some programmers prefer the "home made one-shot" arrangement shown in Figure F because it's based on simple and universally available XIO and OTE instructions. Some programmers feel this makes it easier to transfer one handy "fit all" type program from one PLC platform to another - and even between PLCs from various manufacturers. In many cases this approach works perfectly well - but consider the following scenario. Suppose that SWITCH_A is a limit switch on a conveyor for completed parts. Each time another new part trips the switch, the PLC generates a serial numbered label - which gets applied to the part. Naturally we only want one label for each part - regardless of how many "scans" the PLC processor makes while the switch remains actuated by that part. Suppose that a programmer has used the "home made one-shot" approach shown at the top of Figure F to accomplish this "one-part-one-label" task. Suppose that the system has been working perfectly for a very long time in normal day-to-day operations. Now suppose that while a part is still actuating SWITCH_A, Technician Ted puts the PLC processor into the Program Mode for a few seconds. Putting the system back in the Run Mode will cause the "home made one-shot" to generate an extra pulse of TRUE logic - and trigger an extra labeling operation for the same part. Systems which require careful tracking of serial numbers are highly susceptible to these types of mislabeling and miscounting problems.

(17) In Question 14, notice that both programming methods reacted to a full power cycle by incrementing their ADD instructions. Notice that the processor and the input signal are both electrically powered by the plant's 120VAC service.

(18) In Question 15, notice that both programming methods reacted to a brief power flicker by incrementing their ADD instructions. Notice that the processor and the input signals are both electrically powered by the plant's 120VAC service.

(19) In Question 16, notice that in response to a full power cycle ONLY the "home made one-shot" arrangement triggered its ADD to increment N7:11 - while the OSR did NOT produce a trigger for N7:12. Compare these results to the same full power cycle test in Question 14 where BOTH programming methods incremented their ADD instructions. The difference in operations is caused by using AC power back in Question 14 - and DC power here in Question 16.

(20) In Question 17, notice that in response to a brief power flicker NEITHER programming method produced a trigger to increment its ADD instruction. Compare these results to the same power flicker test in Question 15 where BOTH programming methods incremented their ADD instructions. The difference in operations is caused by using AC power back in Question 15 - and DC power here in Question 17.

(21) Carefully compare the results of "AC powered" Questions 14 and 15 with the results of the identical tests of "DC powered" Questions 16 and 17. Many people, even some with years of experience, fail to realize that using different power sources can cause different results at "Go-To-Run" time - even in cases where the programs themselves are identical. And remember that both of the programming methods shown in Figure F reliably gave identical results when tested in normal day-to-day operations.

Beginner Level Quiz #121 - Further Explanations to the Answers

While the following material isn't intended to be a full "lesson" on all of the concepts involved, we've tried to give each answer enough "bare bones" explanation to help you understand the results of the quiz. Note that we have omitted some details which have no effect on the outcome of the test. If you have questions please feel free to contact us - or else open a topic for discussion on one of the public PLC forums where we contribute regularly.

Question 1 - Answer C. First notice that the two methods did NOT give identical results the way the original programmers assumed they would. **Brief Explanation:** Suppose that the MLX-1000 processor goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A evaluates as FALSE. A TOF must see a TRUE-to-FALSE transition before it will begin timing. No TRUE-to-FALSE transition has occurred - so T4:1 holds its present state with T4:1/DN = 0. On Rung 0001 the XIC for T4:1/DN evaluates as FALSE. FALSE logic causes the OTE to write a value of 0 to the bit/box for PUMP_1. On Rung 0002 the XIO for SWITCH_A evaluates as TRUE. The TON begins timing. T4:2/DN still = 0. On Rung 0003 the XIO for T4:2/DN evaluates as TRUE. TRUE logic causes the OTE to write a value of 1 to the bit/box for PUMP_2. At the end of the first pass through the ladders, PUMP_1 will be OFF, PUMP_2 will be ON. Many scans are made as 5 seconds elapse. Then on the next pass through the ladders, T4:2.ACC = 5, and T4:2/DN = 1. The XIO on Rung 0003 evaluates as FALSE. FALSE logic causes the OTE to write a value of 0 to the bit/box for PUMP_2. At the end of this pass through the ladders, PUMP_2 will be OFF. **Key Point:** The TOF instruction never saw a TRUE-to-FALSE transition so it never began timing. The TON instruction began timing as soon as it saw TRUE logic coming in.

Question 2 - Answer C. First notice that the two methods did NOT give identical results the way the original programmers assumed they would. **Brief Explanation:** When the plant's power is first restored, the system prepares to enter the Run Mode. The MLX-1000 Pre-Scan affects the TOF T4:1 by writing these values: T4:1/EN = 0; T4:1/DN = 0; T4:1/TT = 0; T4:1.ACC = 5. Pre-Scan affects the TON T4:2 by writing these values: T4:2/EN = 0; T4:2/DN = 0; T4:2/TT = 0; T4:2.ACC = 0. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A evaluates as FALSE. A TOF must see a TRUE-to-FALSE transition before it will begin timing. No TRUE-to-FALSE transition has occurred - so T4:1 holds its present state with T4:1/DN = 0. On Rung 0001 the XIC for T4:1/DN evaluates as FALSE. FALSE logic causes the OTE to write a value of 0 to the bit/box for PUMP_1. On Rung 0002 the XIO for SWITCH_A evaluates as TRUE. The TON begins timing. T4:2/DN still = 0. On Rung 0003 the XIO for T4:2/DN evaluates as TRUE. TRUE logic causes the OTE to write a value of 1 to the bit/box for PUMP_2. At the end of the first pass through the ladders, PUMP_1 will be OFF, PUMP_2 will be ON. Many scans are made as 5 seconds elapse. Then on the next pass through the ladders, T4:2.ACC = 5, and T4:2/DN = 1. The XIO on Rung 0003 evaluates as FALSE. FALSE logic causes the OTE to write a value of 0 to the bit/box for PUMP_2. At the end of this pass through the ladders, PUMP_2 will be OFF. **Key Point:** The MLX-1000 Pre-Scan wrote a value of 0 to T4:2.ACC. The TON instruction began timing as soon as it saw TRUE logic coming in.

Question 3 - Answer A. First notice that the results after the plant's power "flickered" were different from the results when the plant's power failed for several minutes. **Brief Explanation:** When the plant's power first dropped off, the MLX-1000 system's internal power supply continued to provide enough power to keep the processor running - and scanning - for a certain amount of time. In most cases, this "hold up" time will be at least a second or two. Since the power was quickly restored, the processor did not leave - and then re-enter - the Run Mode. So no Pre-Scan operation occurred. The TON timer T4:2 was not affected. **Key Point:** The MLX-1000 processor will not normally be affected by a brief power "flicker" that is quickly restored. In simplest terms, as far as the processor knows, the power "flicker" never happened.

Question 4 - Answer C. First notice that the results after a power "flicker" using a PLC-5 system were different from the results using an MLX-1000 system as in the previous question. **Brief Explanation:** When the plant's power first dropped off, the PLC-5 system's power supply provided a signal to the processor to immediately stop scanning and shut down. When the power was restored, the processor re-entered the Run Mode - and a Pre-Scan operation occurred. Pre-Scan affected the TON T4:2 by writing these values: T4:2/EN = 0; T4:2/DN = 0; T4:2/TT = 0; T4:2.ACC = 0. This effectively reset the timer and caused it to "time out" again. **Key Point:** The PLC-5 processor normally WILL be affected by a brief power "flicker" regardless of how quickly the power is restored. In simplest terms, a PLC-5 processor usually treats a power "flicker" the same as a total power "failure". Most other Allen-Bradley platforms act differently.

Question 5 - Answer B. First notice that the two methods did NOT give identical results the way the original programmers assumed they would. **Brief Explanation:** Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The MLX-1000 Pre-Scan affects the CTU C5:3 by writing this value: C5:3/CU = 1. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since the CU bit already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:3 does NOT increment. On Rung 0001 the ADD instruction is executed with TRUE logic. The value stored at N7:4 increments by 1. Many additional scans are made while the value stored at C5:3.ACC remains at 0 - and the value stored at N7:4 continues to rapidly increase. Eventually the value stored at N7:4 reaches 32767. Then on the next pass through the ladders, the ADD instruction attempts to increase the value stored at N7:4 to 32768. This causes a "math overflow" condition and writes a 1 to S:5/0. At the end of this pass through the ladders, S:5/0 will still be ON - which will cause the processor to fault. **Key Point:** The MLX-1000 Pre-Scan wrote a value of 1 to C5:3/CU. The processor then interpreted this as an "already counted" condition - so C5:3.ACC never got incremented. Pre-Scan does not affect an ADD instruction.

Question 6 - Answer F. First notice that the results using a PLC-5 system were different from the results using an MLX-1000 system as in the previous question. **Brief Explanation:** As in the previous question, the value of C5:3.ACC will stay at ZERO throughout the test - and for exactly the same reasons. And the value of N7:4 will rapidly increase to 32767 exactly as before. But the difference is that the PLC-5 platform will NOT automatically generate a fault when a "math overflow" condition occurs - so the value stored at N7:4 will now "roll over" into negative numbers.

There are basically two ways to look at this effect. (1) The PLC-5 is "easier" to work with - since it doesn't automatically "fault" and shut down when a math overflow occurs. (2) The PLC-5 is "trickier" to work with since a math overflow may occur - and not be recognized or acted upon. Specifically, according to the PLC-5 processor, 32767 plus 1 equals NEGATIVE 32768. Now that's certainly not mathematically correct - but the big question is: "Do we want the plant to keep right on running with that incorrect mathematical result?". If the answer is "no" then there ARE programming techniques that we can use to prevent that from happening. And conversely, if we DO want the MLX-1000 system to continue running in spite of a "math overflow" event we CAN program that to happen. **Key Point:** The same program can function differently when tested under identical conditions if different PLC platforms are involved.

Question 7 - Answer D. First notice that the two methods did NOT give identical results the way the original programmers assumed they would. **Brief Explanation:** Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The MLX-1000 Pre-Scan affects the CTU C5:5 by writing this value: C5:5/CU = 1. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the "First Pass" bit S:1/15 = 1; so the XIC for the "First Pass" bit is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since the CU bit already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:5 does NOT increment. On Rung 0001 the "First Pass" bit S:1/15 = 1; so the XIC for the "First Pass" bit is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:6 increments by 1; so 0 plus 1 equals 1. Now consider the next scan through the ladders. On Rung 0000 the "First Pass" bit S:1/15 = 0; so the XIC for the "First Pass" bit is evaluated as FALSE. The CTU instruction is executed with FALSE logic. The processor writes a value of 0 to C5:5/CU bit. On Rung 0001 the "First Pass" bit S:1/15 = 0; so the XIC for the "First Pass" bit is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:6 does NOT increment by 1. Eventually the system is placed in the Program Mode. Then the system is placed back in the Run Mode again. What happens next should sound familiar. Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The MLX-1000 Pre-Scan affects the CTU C5:5 by writing this value: C5:5/CU = 1. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the "First Pass" bit S:1/15 = 1; so the XIC for the "First Pass" bit is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since the CU bit already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:5 does NOT increment. On Rung 0001 the "First Pass" bit S:1/15 = 1; so the XIC for the "First Pass" bit is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:6 increments by 1; so 1 plus 1 equals 2. Now consider the next scan through the ladders. On Rung 0000 the "First Pass" bit S:1/15 = 0; so the XIC for the "First Pass" bit is evaluated as FALSE. The CTU instruction is executed with FALSE logic. The processor writes a value of 0 to C5:5/CU bit. On Rung 0001 the "First Pass" bit S:1/15 = 0; so the XIC for the "First Pass" bit is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:6 does NOT increment by 1.

Key Point: The ADD instruction makes an accurate count of the two "First Pass" conditions. On the other hand, Pre-Scan prevents the CTU from counting the "First Pass" events. Also, don't miss the CRITICAL concept that Pre-Scan happens FIRST and THEN the "First Pass" is made. Specifically, Pre-Scan and "First Pass" are NOT the same thing. Many programmers (both beginners and those with some experience) confuse the two operations.

Question 8 - Answer B. First notice that the results are different from the similar programming methods used in the previous question. **Brief Explanation:** Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The MLX-1000 Pre-Scan does NOT affect the OSR instruction on Rung 0000. Since the question specifies that this is a "brand new" system, the bit B3/7 contains a "fresh-out-of-the-box" 0. The Pre-Scan affects the CTU C5:7 by writing this value: C5:7/CU = 1. Pre-Scan does NOT affect the OSR instruction on Rung 0001. Since this is a "new" program, the bit B3/8 contains a "fresh-out-of-the-box" 0. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the processor checks the status of bit B3/7 and finds a 0. This tells the processor that the OSR has NOT already been "fired"; so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/7 to mark that the OSR has now been "fired". The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since the CU bit already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:7 does NOT increment. On Rung 0001 the processor checks the status of bit B3/8 and finds a 0. This tells the processor that the OSR has NOT already been "fired"; so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/8 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:8 increments by 1; so 0 plus 1 equals 1. Now consider the next scan through the ladders. On Rung 0000 the processor checks the status of bit B3/7 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The CTU instruction is executed with FALSE logic. The processor writes a value of 0 to C5:7/CU. On Rung 0001 the processor checks the status of bit B3/8 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:8 does NOT increment by 1. Eventually the system is placed in the Program Mode. Then the system is placed back in the Run Mode again. What happens next should sound familiar - but stay on your toes. Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The MLX-1000 Pre-Scan does NOT affect the OSR instruction on Rung 0000. Bit B3/7 still contains a 1. The Pre-Scan affects the CTU C5:7 by writing this value: C5:7/CU = 1. Pre-Scan does NOT affect the OSR instruction on Rung 0001. The bit B3/8 still contains a 1. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the processor checks the status of bit B3/7 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The CTU instruction is executed with FALSE logic. The processor writes a value of 0 to C5:7/CU. On Rung 0001 the processor checks the status of bit B3/8 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:8 does NOT increment by 1; so the value of 1 stays at 1. **Key Point:** In an MLX-1000 system, Pre-Scan does NOT affect an OSR instruction. This can lead to surprises when a "brand new, fresh out of the box" program works OK the FIRST time that it EVER gets tested - but then never works right again.

Question 9 - Answer A. First notice that neither of the two "counting" methods were incremented when the processor went from Run Mode - to Program Mode - and back to Run Mode - even though the "input" condition controlling each rung stayed TRUE throughout the test. **Brief Explanation:** The question specifies that the test begins with the MLX-1000 processor in the Run Mode. SWITCH_A in the field is ON. Here is what is happening on each pass as the processor executes the ladder logic. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The question specifies that C5:9.ACC contains the value 0 - therefore any previous count must have already been (manually) cleared out. C5:9/CU must already contain a 1 or else C5:9 would increment based on the TRUE logic - but it does NOT increment. Note that TRUE logic wrote the value of 1 to C5:9/CU on a previous scan. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. Now the processor is placed in the Program Mode. Then the processor is placed back in the Run Mode. Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The Pre-Scan affects the CTU by writing this value: C5:9/CU = 1. Pre-Scan does NOT affect the OSR instruction on Rung 0001. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since bit C5:9/CU already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:9 does NOT increment. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. **Key Point:** The CTU instruction did NOT increment when the processor went into the Run Mode because Pre-Scan wrote a value of 1 into C5:9/CU. The ADD instruction did NOT increment when the processor went into the Run Mode - even though Pre-Scan does NOT affect the OSR instruction. So both "counter" programming methods gave the same result in this particular test - but for two different reasons.

Question 10 - Answer C. First notice that the two programming methods gave different results even though they were both being controlled by the same "input" condition from the field. **Brief Explanation:** The question specifies that the test begins with the MLX-1000 processor in the Run Mode. SWITCH_A in the field is OFF. Here is what is happening on each pass as the processor executes the ladder logic. On Rung 0000 the XIC for SWITCH_A is evaluated as FALSE. The CTU instruction is executed with FALSE logic. FALSE logic writes the value of 0 to C5:9/CU. On Rung 0001 the XIC for SWITCH_A is evaluated as FALSE. The OSR is executed with FALSE logic. The processor writes a value of 0 to bit B3/10 to "reload" the OSR. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. Now the processor is placed in the Program Mode. SWITCH_A in the field is turned ON. Now the processor is placed back in the Run Mode. Before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The Pre-Scan affects the CTU by writing this value: C5:9/CU = 1. Pre-Scan does NOT affect the OSR instruction on Rung 0001. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic.

The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since bit C5:9/CU already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:9 does NOT increment. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 0. This tells the processor that the OSR has NOT already been "fired" - so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/10 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:10 increments by 1. **Key Point:** The CTU didn't increment when the processor went into the Run Mode because Pre-Scan had written a value of 1 into C5:9/CU. The ADD instruction DID increment when the processor went into the Run Mode because Pre-Scan does NOT affect the OSR instruction - and the OSR had been "reloaded" by FALSE logic on a previous scan.

Question 11 - Answer C. First notice that the two methods did NOT give identical results the way the original programmers assumed they would. **Brief Explanation:** Here is what happens on each pass through the ladders just before the plant's power fails. The MLX-1000 processor is in the Run Mode. SWITCH_A in the field is ON. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. Since C5:9/CU already contains a value of 1, it appears that no FALSE-to-TRUE transition has taken place - so C5:9 does NOT increment. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. Now the plant's power failed for several minutes. As soon as the plant's power first dropped off, the 120VAC line power for the input signal from SWITCH_A instantly dropped to zero volts. But the MLX-1000 system's internal power supply continued to provide enough power to keep the processor running - and scanning - for a certain amount of time. During this "hold up" period, the processor "saw" the electrical input from SWITCH_A appear to be an OFF signal - even though SWITCH_A in the field physically stayed in the ON position. Here is what happens on each pass through the ladders during the brief "hold up" period just after the plant's power fails. On Rung 0000 the XIC for SWITCH_A is evaluated as FALSE. The CTU instruction is executed with FALSE logic. FALSE logic writes a value of 0 to C5:9/CU. On Rung 0001 the XIC for SWITCH_A is evaluated as FALSE. The OSR is executed with FALSE logic. FALSE logic writes the value of 0 to bit B3/10 to "reload" the OSR. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. Then the plant's power stayed OFF for several minutes. And then the plant's power came back ON. Just before the system actually enters the Run Mode, the processor does a Pre-Scan operation. The Pre-Scan affects the CTU by writing this value: C5:9/CU = 1. Pre-Scan does NOT affect the OSR instruction on Rung 0001. Pre-Scan does NOT affect the ADD instruction. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since bit C5:9/CU already contains a 1, it appears that no FALSE-to-TRUE transition has occurred - so C5:9 does NOT increment. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 0. This tells the processor that the OSR has NOT already been "fired" - so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/10 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:10 increments by 1.

Key Point: The CTU instruction did NOT increment when the processor went back into the Run Mode because Pre-Scan had written a value of 1 into C5:9/CU. The ADD instruction DID increment when the processor went into the Run Mode because the OSR had been "reloaded" by the apparent OFF signal from SWITCH_A during the processor's "hold up" period immediately after the plant's power failed.

Question 12 - Answer D. First notice that the results of this "power flicker" test are different from the results of the "power failure" test in the previous question. **Brief Explanation:** Here is what happens on each pass through the ladders just before the plant's power "flickers". The MLX-1000 processor is in the Run Mode. SWITCH_A in the field is ON. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. Since C5:9/CU already contains a value of 1, it appears that no FALSE-to-TRUE transition has taken place - so C5:9 does NOT increment. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. Now the plant's power "flickers" OFF for a fraction of a second. As soon as the plant's power first dropped off, the 120VAC line power for the input signal from SWITCH_A instantly dropped to zero volts. But the MLX-1000 system's internal power supply continued to provide enough power to keep the processor running - and scanning - for a certain amount of time. During this "hold up" period, the processor "saw" the electrical input from SWITCH_A appear to be an OFF signal - even though SWITCH_A in the field physically stayed in the ON position. Here is what happens on each pass through the ladders during the brief "hold up" period just after the plant's power "flickers" OFF. On Rung 0000 the XIC for SWITCH_A is evaluated as FALSE. The CTU instruction is executed with FALSE logic. FALSE logic writes a value of 0 to C5:9/CU. On Rung 0001 the XIC for SWITCH_A is evaluated as FALSE. The OSR is executed with FALSE logic. FALSE logic writes the value of 0 to bit B3/10 to "reload" the OSR. The ADD instruction is executed with FALSE logic. The value stored at N7:10 does NOT increment. And then the plant's power quickly came back ON. Since the processor's "hold up" time had not expired, the processor never really "shut down". Specifically, the processor didn't "see" its internal operating power go OFF at all - so in this test the system does NOT go through a Pre-Scan operation. More specifically, the processor simply stayed in the Run Mode while the power briefly "flickered" OFF. Here is what happens on the next pass through the ladders just after the plant's power comes back ON again. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. The CTU instruction is executed with TRUE logic. The processor checks the counter's CU bit to see if a FALSE-to-TRUE transition has occurred. Since bit C5:9/CU contains a 0, it appears that a FALSE-to-TRUE transition HAS occurred - so C5:9 DOES increment. TRUE logic writes a value of 1 to C5:9/CU to mark that a "count" has been made. On Rung 0001 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/10 and finds a 0. This tells the processor that the OSR has NOT already been "fired" - so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/10 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:10 increments by 1. **Key Point:** The CTU instruction DID increment in this test because the brief loss of power made the electrical input signal from SWITCH_A transition from TRUE to FALSE and back to TRUE. The same thing happened in the previous "power failure" question - but this time no Pre-Scan operation took place since the power "flicker" wasn't OFF long enough to "restart" the processor.

Question 13 - Answer B. First notice that the two methods did NOT give identical results the way the original programmers assumed they would.

Brief Explanation: Here is what happens on each pass through the ladders before going to the Program Mode. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Now the processor is placed in the Program Mode. SWITCH_A in the field remains ON. Now the processor is placed back in the Run Mode. Before the system restarts and enters the Run Mode, the processor does a Pre-Scan operation. Pre-Scan affects the OTE for bit B3/11 - by writing a value of 0 to bit B3/11. Pre-Scan affects the OTE for bit B3/32 - by writing a value of 0 to bit B3/32. Pre-Scan does NOT affect the ADD instruction on Rung 0001. Pre-Scan does NOT affect the OSR instruction on Rung 0002. Pre-Scan does NOT affect the ADD instruction on Rung 0002. Notice that at this point in time, Andy's "homemade one-shot" has been "reloaded" - but Bert's OSR has NOT. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 0. The XIO for B3/32 is evaluated as TRUE. The OTE for B3/11 is executed with TRUE logic - and writes the value of 1 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:11 DOES increment; so 0 plus 1 equals 1. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Here's what happens on the next (and subsequent) passes through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. **Key Point:** Pre-Scan's effect on the OTE instruction for B3/32 "reloaded" the "homemade one-shot" arrangement. Later, when the processor went back into the Run Mode, the "homemade one-shot" caused the ADD for N7:11 to increment its value. Since Pre-Scan does not affect the OSR instruction, the ADD for N7:12 did NOT increment.

Question 14 - Answer D. First notice that both of the programming methods DID give a "count" after the plant's 120VAC power failed for several minutes. **Brief Explanation:** Here is what happens on each pass through the ladders before the power failure. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Now the plant's 120VAC power fails for several minutes. As soon as the plant's power first dropped off, the 120VAC line power for the input signal from SWITCH_A instantly dropped to zero volts. But the MLX-1000 system's internal power supply continued to provide enough power to keep the processor running - and scanning - for a certain amount of time. During this "hold up" period, the processor "saw" the electrical input from SWITCH_A appear to be an OFF signal - even though SWITCH_A in the field physically stayed in the ON position. Here is what happens on each pass through the ladders during the brief "hold up" period just after the plant's power fails. On Rung 0000 the XIC for SWITCH_A is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with FALSE logic - and writes the value of 0 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as FALSE. The OSR is executed with FALSE logic. FALSE logic writes the value of 0 to bit B3/12 to "reload" the OSR. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. In simplest terms, Andy's "home made one-shot" and Bert's OSR have both been "reloaded" at this point in time. The "hold up" period of the processor's built-in power supply finally expires and the processor stops scanning. SWITCH_A in the field remains ON. Minutes later the AC power is restored. Before the system restarts and enters the Run Mode, the processor does a Pre-Scan operation. Pre-Scan affects the OTE for bit B3/11 - by writing a value of 0 to bit B3/11. Pre-Scan affects the OTE for bit B3/32 - by writing a value of 0 to bit B3/32. Pre-Scan does NOT affect the ADD instruction on Rung 0001. Pre-Scan does NOT affect the OSR instruction on Rung 0002. Pre-Scan does NOT affect the ADD instruction on Rung 0002. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 0. The XIO for B3/32 is evaluated as TRUE. The OTE for B3/11 is executed with TRUE logic - and writes the value of 1 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:11 DOES increment; so 0 plus 1 equals 1. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 0. This tells the processor that the OSR has NOT already been "fired" - so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/12 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:12 DOES increment; so 0 plus 1 equals 1.

Here's what happens on the next (and subsequent) passes through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. **Key Point:** Using 120VAC power for the system, even though SWITCH_A in the field remained physically ON throughout the test, the electrical input signal from SWITCH_A appeared to go OFF while the processor kept scanning during its "hold up" time. This apparent OFF signal "reloaded" both of the "one-shot" arrangements. When the power came back ON, both of the "one-shot" arrangements "fired". Each ADD instruction received one pulse of TRUE logic and incremented its value.

Question 15 - Answer D. First notice that both of the programming methods DID give a "count" after the plant's 120VAC power "flickered" for a fraction of a second. **Brief Explanation:** Here is what happens on each pass through the ladders before the power "flickers". On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Now the plant's 120VAC power "flickers" OFF for a fraction of a second. As soon as the plant's power first dropped off, the 120VAC line power for the input signal from SWITCH_A instantly dropped to zero volts. But the MLX-1000 system's internal power supply continued to provide enough power to keep the processor running - and scanning - for a certain amount of time. During this "hold up" period, the processor "saw" the electrical input from SWITCH_A appear to be an OFF signal - even though SWITCH_A in the field physically stayed in the ON position. Here is what happens on each pass through the ladders during the brief "hold up" period while the plant's power is "flickered" OFF. On Rung 0000 the XIC for SWITCH_A is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with FALSE logic - and writes the value of 0 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as FALSE. The OSR is executed with FALSE logic. FALSE logic writes the value of 0 to bit B3/12 to "reload" the OSR. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. In simplest terms, Andy's "home made one-shot" and Bert's OSR have both been "reloaded" at this point in time. Now before the "hold up" period of the processor's built-in power supply expires, the plant's 120VAC power is restored. Notice that the processor never fully "shut down" so in this particular "flicker" test a Pre-Scan operation will NOT be done.

The processor stays in the Run Mode and makes its next pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 0. The XIO for B3/32 is evaluated as TRUE. The OTE for B3/11 is executed with TRUE logic - and writes the value of 1 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:11 DOES increment; so 0 plus 1 equals 1. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 0. This tells the processor that the OSR has NOT already been "fired" - so the OSR is evaluated as TRUE. The processor writes a value of 1 to bit B3/12 to mark that the OSR has now been "fired". The ADD instruction is executed with TRUE logic. The value stored at N7:12 DOES increment; so 0 plus 1 equals 1. Here's what happens on the next (and subsequent) passes through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. **Key Point:** Using 120VAC power for the system, even though SWITCH_A in the field physically remained ON throughout the test, when the power "flickered" OFF, the electrical input signal from SWITCH_A appeared to go OFF while the processor kept scanning during its "hold up" time. This apparent OFF signal "reloaded" both of the "one-shot" arrangements. When the power came back ON, both of the "one-shot" arrangements "fired". Each of the ADD instructions received one pulse of TRUE logic - and both of them incremented their values.

Question 16 - Answer B. First notice that after the plant's power failed for several minutes, the "homemade one-shot" arrangement DID give a "count" - but the OSR arrangement did NOT. This outcome is different from the results we got from our previous "power failure" test back in Question 14. In that test N7:12 DID count - but here it doesn't. Remember that we're using Schematic 3 this time - and so both the MLX-1000 processor and SWITCH_A in the field are being fed from an external 24VDC power supply. That's the tricky part. **Brief Explanation:** Here is what happens on each pass through the ladders before the power failure. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment.

Now the plant's 120VAC power fails for several minutes. Even though the plant's 120VAC line power instantly dropped off to zero volts, the external DC power supply continued to feed power to the processor and to SWITCH_A in the field. That's the tricky part. Specifically, whenever the plant's power "fails", an AC power source will drop to zero volts INSTANTLY - but with most DC power supplies the voltage dies off GRADUALLY. The timing is important. In the case of our test system, the processor shuts down and quits scanning the ladders BEFORE the input signal from SWITCH_A drops out. Unlike our previous tests, in this case the processor did NOT "see" the electrical input from SWITCH_A appear to be an OFF signal. In simplest terms, the processor quits scanning before it "misses" the electrical input signal from SWITCH_A in the field. Said another way, as the processor "goes to sleep" it still considers Andy's "home made one-shot" and Bert's OSR to both have already been "fired". More specifically, at this point in time neither "one-shot" arrangement has been "reloaded". SWITCH_A in the field remains ON. Minutes later the plant's 120VAC power is restored - and the 24VDC power supply begins feeding both the processor and SWITCH_A in the field. Before the system restarts and enters the Run Mode, the processor does a Pre-Scan operation. Pre-Scan affects the OTE for bit B3/11 - by writing a value of 0 to bit B3/11. Pre-Scan affects the OTE for bit B3/32 - by writing a value of 0 to bit B3/32. Pre-Scan does NOT affect the ADD instruction on Rung 0001. Pre-Scan does NOT affect the OSR instruction on Rung 0002. Pre-Scan does NOT affect the ADD instruction on Rung 0002. Notice that at this point in time, Andy's "homemade one-shot" has been "reloaded" - but Bert's OSR has NOT. The processor then goes into the Run Mode and makes its first pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 0. The XIO for B3/32 is evaluated as TRUE. The OTE for B3/11 is executed with TRUE logic - and writes the value of 1 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as TRUE. The ADD instruction is executed with TRUE logic. The value stored at N7:11 DOES increment; so 0 plus 1 equals 1. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Here's what happens on the next (and subsequent) passes through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. **Key Point:** Using a 24VDC power supply allowed the electrical input signal from SWITCH_A in the field to remain ON while the processor was shutting down. This prevented either of the "one-shot" arrangements from being "reloaded" in the same way that they were in the two previous questions. But in spite of this, the "homemade one-shot" arrangement DID get "reloaded" anyway due to the Pre-Scan's effect on the OTE instructions. So only the first ADD instruction received a pulse of TRUE logic and incremented its value.

Question 17 - Answer A. First notice that NEITHER of the programming methods gave a "count" after the plant's power "flickered" for a fraction of a second. This outcome is different from the results we got from our previous "power flicker" test back in Question 15. In that test, BOTH N7:11 and N7:12 counted - here NEITHER of them do. Remember we're using Schematic 3 in this test. Notice that both the MLX-1000 processor and SWITCH_A in the field are being fed from an external 24VDC power supply. That's the tricky part. **Brief Explanation:** Here is what happens on each pass through the ladders before the power "flickers". On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. Now the plant's 120VAC power "flickers" OFF for a fraction of a second. Even though the plant's 120VAC line power instantly drops off to zero volts, the external DC power supply continues to feed power to the processor and to SWITCH_A in the field. Unlike our previous "flicker" test (with 120VAC power to the I/O circuit), in this case the processor continues to "see" the electrical input from SWITCH_A in the field as a constant ON signal. In simplest terms, the processor never "misses" the electrical input signal from SWITCH_A in the field. Said another way, the processor still considers Andy's "home made one-shot" and Bert's OSR to both have already been "fired". More specifically, neither "one-shot" arrangement has been "reloaded". A fraction of a second later the plant's 120VAC power is restored - and the 24VDC power supply continues feeding both the processor and SWITCH_A in the field. Notice that the processor never fully "shut down" so in this particular "flicker" test a Pre-Scan operation will NOT be done. The processor stays in the Run Mode and makes its next pass through the ladders. On Rung 0000 the XIC for SWITCH_A is evaluated as TRUE. Bit B3/32 contains a value of 1. The XIO for B3/32 is evaluated as FALSE. The OTE for B3/11 is executed with FALSE logic - and writes the value of 0 to bit B3/11. The OTE for B3/32 is executed with TRUE logic - and writes the value of 1 to bit B3/32. On Rung 0001 the XIC for bit B3/11 is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:11 does NOT increment. On Rung 0002 the XIC for SWITCH_A is evaluated as TRUE. The OSR is executed with TRUE logic. The processor checks the status of bit B3/12 and finds a 1. This tells the processor that the OSR HAS already been "fired" - so the OSR is evaluated as FALSE. The ADD instruction is executed with FALSE logic. The value stored at N7:12 does NOT increment. **Key Point:** Using a 24VDC power supply allowed both the processor and the electrical input signal from SWITCH_A in the field to remain ON while the plant's power "flickered". Neither of the "one-shot" arrangements got "reloaded" - and the processor did NOT shut down and require a Pre-Scan operation. Neither one of the ADD instructions incremented its value.

As we said earlier, while the material above isn't intended to be a full "lesson" on all of the concepts involved, we've tried to give each answer enough explanation to help you understand the results of the quiz. If you have additional questions, feel free to contact us.

Discussion on the Basic Concepts Involved in Quiz #121

The main concepts that we set out to demonstrate in this quiz are listed below:

- (1) Certain programming techniques may give a particular result when tested in normal day-to-day operations - and yet give radically different results at "Go-To-Run" time.
- (2) Certain programming techniques may give a particular result when tested the first time - and yet never give the same result again.
- (3) Certain programming techniques may give a particular result when the main power fails completely - and yet give another result when the power only flickers for a fraction of a second.
- (4) Certain programming techniques may give a particular result when the system is powered by AC line voltage - and yet give another result when powered by a DC power supply.
- (5) Certain programming techniques may give a particular result when used on one PLC platform - and yet give another result when used on another platform - even one from the same manufacturer.
- (6) Certain programming techniques may give a particular result when field input signals are in one state at "start up" or "shut down" - and yet give another result when the inputs are in another state the next time the system is started or stopped.

All of the concepts that we've demonstrated in this quiz have been related (at least in some fashion) to what might be called "Go-To-Run" situations. Over the years we've had many people tell us that they've been unpleasantly surprised from time-to-time by the types of "gremlins" that we've covered here. The simple truth is that very few programmers fully understand these issues - and therefore fail to consider their effects on the systems that are being controlled. Various problems can arise when concepts like these aren't taken into account. The most challenging aspect of these types of problems is that they appear to be so intermittent and random in nature. This, of course, makes them exceptionally hard to troubleshoot and correct.

As we said at the beginning, this quiz is meant to be entertaining and thought-provoking - and hopefully educational as well. We certainly haven't exhausted all of the concepts available for discussion along these lines, but we've tried to at least introduce some of the most common culprits to "look out for" when programming, debugging, troubleshooting, and operating PLC-controlled systems in the field.

Naturally the simple "count" and "increment" tests used in this quiz are just illustrations of how certain programming techniques may give brief "pulses" of TRUE logic at unexpected times - especially when "Go-To-Run" situations are involved. In many cases such pulses may have no effect on the system's operation. In other cases the results may be quite serious.

We strongly recommend that you spend some time reviewing the material in this quiz (#121 "Surprises at Go-To-Run") before moving on to the next one in the series (#122 "More About Go-To-Run") which continues along the same lines with questions that most readers will consider to be slightly more challenging.

Beginner Level Quiz #121 - Summing Up

One of the main objectives of all of our Email Quizzes is to expose some of the common misconceptions that many PLC technicians believe to be true. Unfortunately much of this material "sounds right" even though it happens to be totally wrong. The fact that these wrong ideas are so commonly believed and so widely circulated helps explain why some people find PLC skills so hard to master. If you're interested in how our PLC Boot Camp classes are specifically designed to weed out and correct these types of mistakes and misconceptions, you can find a lot of detailed information on our website at www.ronbeaufort.com.

We hope that you've enjoyed the quiz - and possibly learned something from it. We've tried to keep the test conditions in our experiments as close as possible to the "normal" conditions that you'll run into while working in the field. But, as they say: "Your mileage may vary." Just remember that there are many factors which can affect the results of any test. That's why it's always best to learn and understand the "nuts and bolts" of why each PLC system acts the way it does - instead of relying on popular "rules of thumb" which might not be accurate in every situation.

If you'd like to discuss any of the material in our PLC Quiz, just contact us and give us a chance to go over it with you. We'll be glad to answer any questions that you might have. You might also want to follow some of the "Links to PLC Resources" on our website for one of the public PLC-related internet forums where we make frequent contributions. You're welcome to start your own thread there and post your questions about any of our quizzes online. Registration is free and forum members from around the world will be glad to help you understand the concepts involved.

QUIZ #121 COMPARISON MATRIX

QUESTION AND ANSWER	QUES	ANS	PLC PLATFORM	WIRING SCHEMATIC	NOTES ON TEST CONDITIONS	TEST SHORTHAND	RESULT AT TOP	RESULT AT BOTTOM	NOTES ON TEST RESULTS
PUMP_1 DOES NOT RUN PUMP_2 RUNS 5 SECONDS	1	C	MLX-1000	SCHM 1	PROGRAM MODE - SWITCH OFF RUN MODE - SWITCH OFF	P R 0 0	—	⌋	
PUMP_1 DOES NOT RUN PUMP_2 RUNS 5 SECONDS	2	C	MLX-1000	SCHM 1	POWER OK - SWITCH OFF POWER FAILS - SWITCH OFF POWER RESTORED - SWITCH OFF	0 0	—	⌋	
PUMP_1 DOES NOT RUN PUMP_2 DOES NOT RUN	3	A	MLX-1000	SCHM 1	POWER OK - SWITCH OFF POWER FLICKERS - SWITCH OFF POWER OK - SWITCH OFF	0 0	—	—	
PUMP_1 DOES NOT RUN PUMP_2 RUNS 5 SECONDS	4	C	PLC-5	SCHM 1	POWER OK - SWITCH OFF POWER FLICKERS - SWITCH OFF POWER OK - SWITCH OFF	0 0	—	⌋	
C5:3 DOES NOT COUNT N7:4 COUNTS UP AND FAULTS	5	B	MLX-1000	SCHM 1	PROGRAM MODE RUN MODE	P R	0	FAULT	
C5:3 DOES NOT COUNT N7:4 COUNTS AND ROLLS OVER	6	F	PLC-5	SCHM 1	PROGRAM MODE RUN MODE	P R	0	⌋	
C5:5 DOES NOT COUNT N7:6 COUNTS UP TWICE	7	D	MLX-1000	SCHM 1	PROGRAM MODE - RUN MODE PROGRAM MODE - RUN MODE	P R P R	0	2	
C5:7 DOES NOT COUNT N7:8 COUNTS UP ONCE	8	B	MLX-1000	SCHM 1	PROGRAM MODE - RUN MODE PROGRAM MODE - RUN MODE	P R P R	0	1	
C5:9 DOES NOT COUNT N7:10 DOES NOT COUNT	9	A	MLX-1000	SCHM 1	RUN MODE - SWITCH ON PROGRAM MODE - SWITCH ON RUN MODE - SWITCH ON	R P R 1 1 1	0	0	
C5:9 DOES NOT COUNT N7:10 DOES COUNT	10	C	MLX-1000	SCHM 1	RUN MODE - SWITCH OFF PROGRAM MODE - SWITCH ON RUN MODE - SWITCH ON	R P R 1 0 1	0	1	
C5:9 DOES NOT COUNT N7:10 DOES COUNT	11	C	MLX-1000	SCHM 1	POWER OK - SWITCH ON POWER FAILS - SWITCH ON POWER RESTORED - SWITCH ON	1 1	0	1	
C5:9 DOES COUNT N7:10 DOES COUNT	12	D	MLX-1000	SCHM 1	POWER OK - SWITCH ON POWER FLICKERS - SWITCH ON POWER OK - SWITCH ON	1 1	1	1	
N7:11 DOES COUNT N7:12 DOES NOT COUNT	13	B	MLX-1000	SCHM 1	RUN MODE - SWITCH ON PROGRAM MODE - SWITCH ON RUN MODE - SWITCH ON	R P R 1 1 1	1	0	
N7:11 DOES COUNT N7:12 DOES COUNT	14	D	MLX-1000	SCHM 1	POWER OK - SWITCH ON POWER FAILS - SWITCH ON POWER RESTORED - SWITCH ON	1 1	1	1	
N7:11 DOES COUNT N7:12 DOES COUNT	15	D	MLX-1000	SCHM 1	POWER OK - SWITCH ON POWER FLICKERS - SWITCH ON POWER OK - SWITCH ON	1 1	1	1	
N7:11 DOES COUNT N7:12 DOES NOT COUNT	16	B	MLX-1000	SCHM 3	POWER OK - SWITCH ON POWER FAILS - SWITCH ON POWER RESTORED - SWITCH ON	1 1	1	0	
N7:11 DOES NOT COUNT N7:12 DOES NOT COUNT	17	A	MLX-1000	SCHM 3	POWER OK - SWITCH ON POWER FLICKERS - SWITCH ON POWER OK - SWITCH ON	1 1	0	0	