PLS - Kombinatoriske oppgaver

Kompetansemål:

- planlegge, programmere, montere og idriftsette programmerbare styresystemer
- endre og tilpasse skjermbilder for grensesnitt mellom menneske og maskin
- anvende ulike elektroniske kommunikasjonssystemer i automatiserte anlegg

Læringsmål

- Kunne løse kombinatoriske styringer med PLS programmering

Forkunnskaper

-

Teori

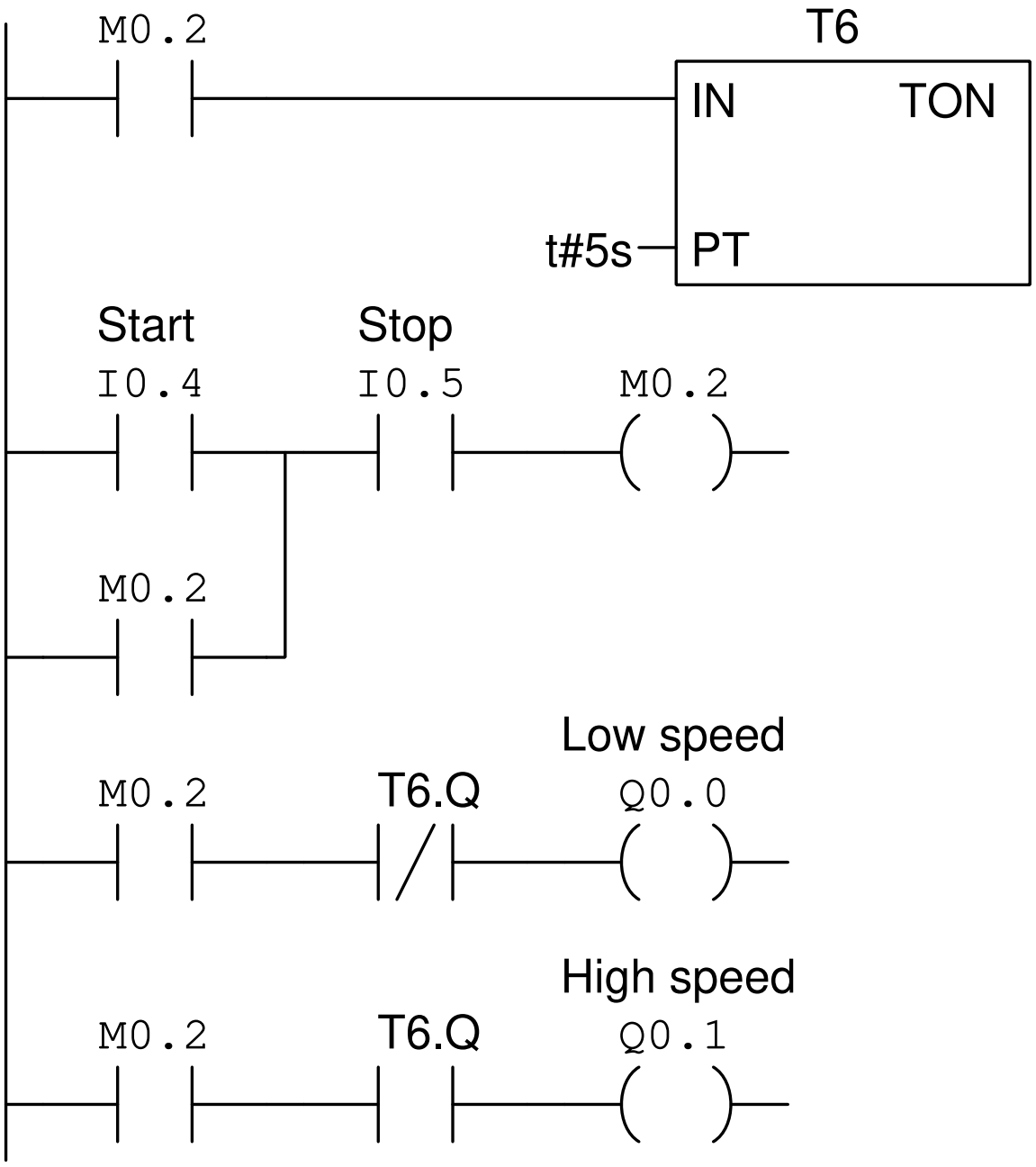
Øvingsoppgaver til leksjon - følger neste side

Innlevering til leksjon - Det er ingen innlevering til leksjonen.
Under disse oppgavene skal du ha fokus på følgende regler fra coding guidelines:
4.2 All elements shall be commented
4.1 Comment shall describe the intention of the code
5.3 All variables shall be initialized before being used.
5.13 Physical outputs shall be written once per PLC cycle
5.8. Floating point comparison shall not be equality or inequality
5.9. Time and physical measures comparison shall not be equality or inequality

Tilkoblet utstyr	IO på PLS	Variabel	Beskrivelse av tilkoblet utstyr
S1	DI1	Start	Start av motor
S2	DI2	Stopp	Stopp av motor
F4	DI3	Motorvern	Inngang for motorvern
Q1	DO1	Motor	Utgang for styring av motor
H1	DO2	Varsellys	Varsel for motorvern utløst.

Oppgave 1

Analyze this Siemens S7-200 PLC program (for controlling a motor) and explain what it is supposed to do:

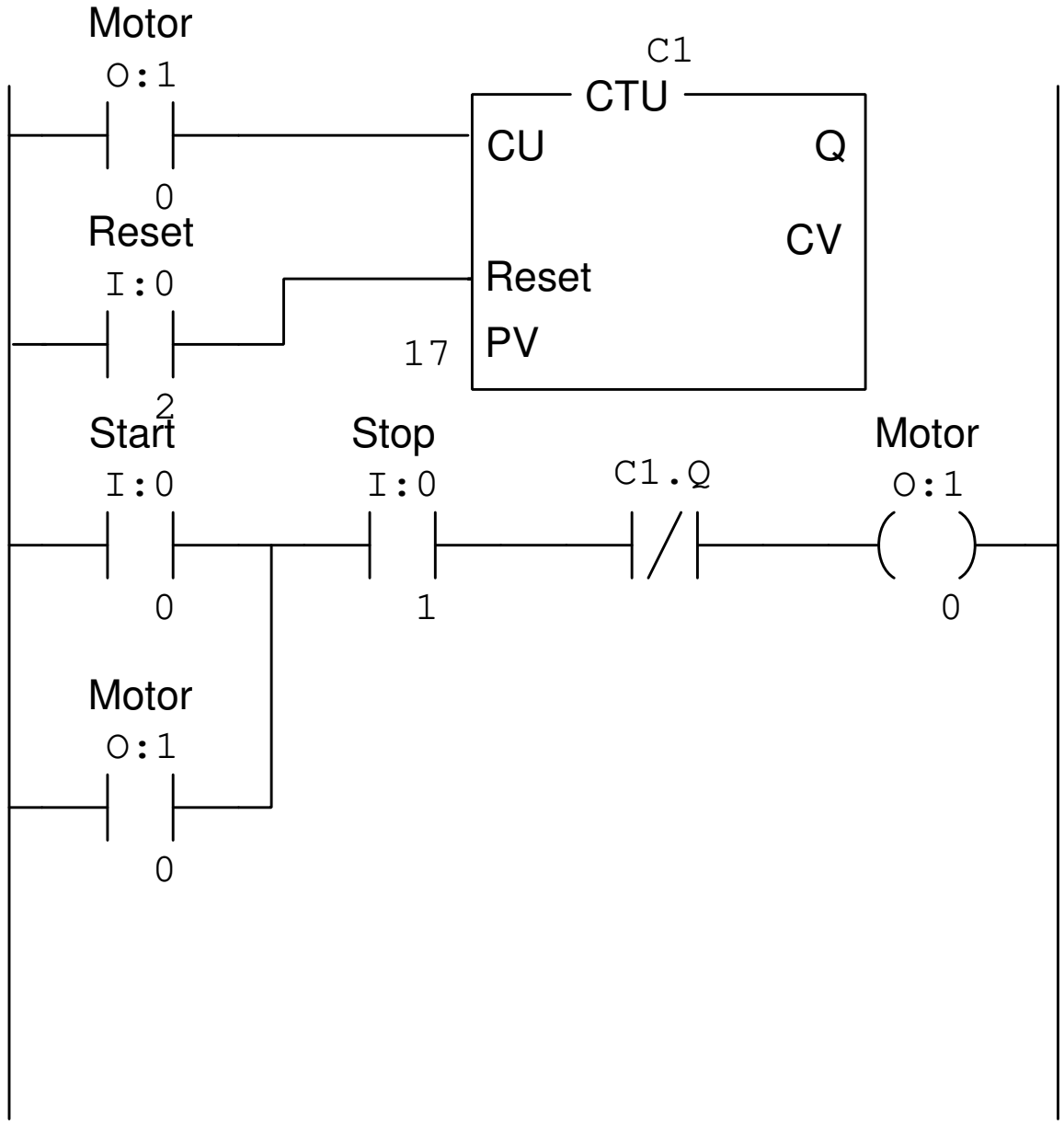


Include an explanation of the motor contactor wiring, based on an analysis of the PLC program.

file i02255

Oppgave 2

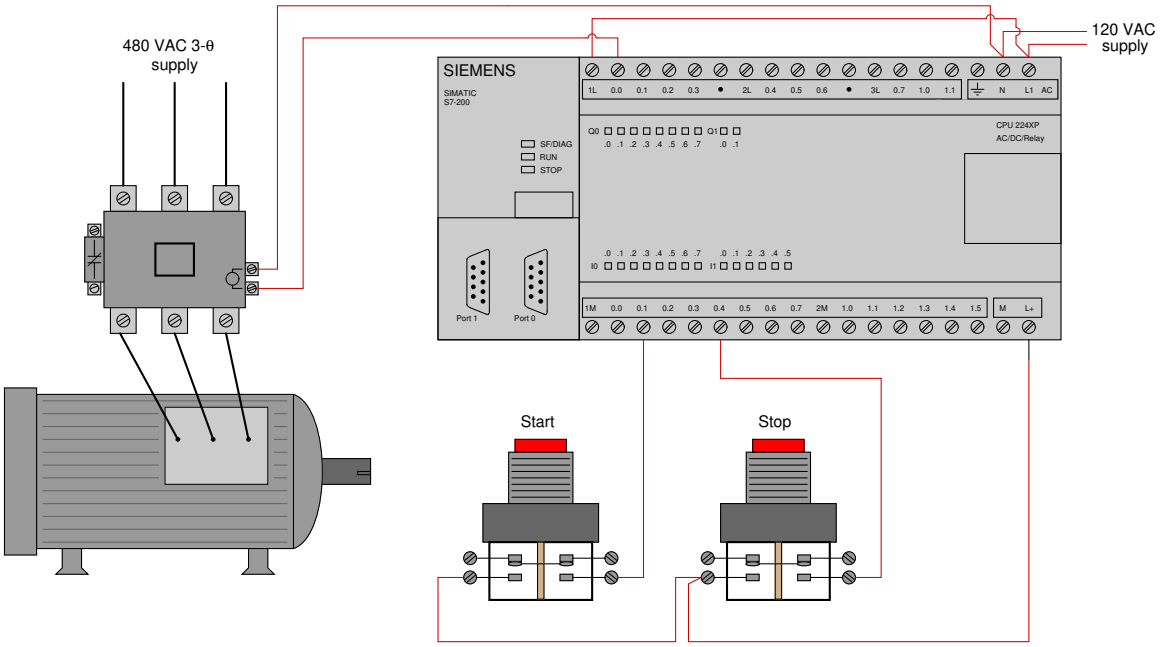
Analyze this Allen-Bradley PLC program and explain what it is supposed to do:



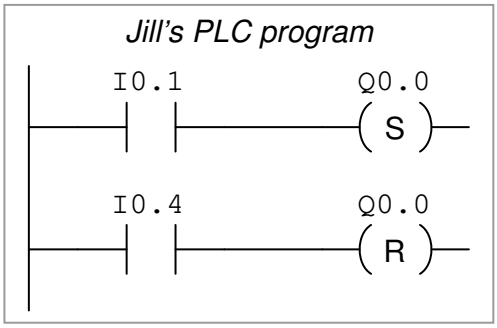
file i02377

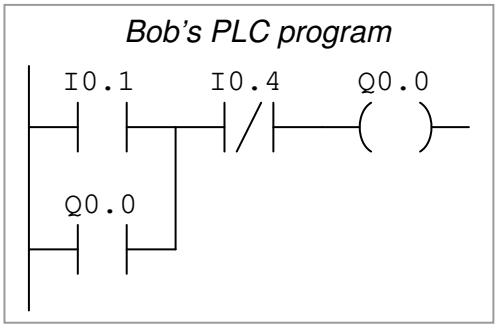
Oppgave 3

Two technicians, Jill and Bob, work on programming Siemens S7-200 PLCs to control the starting and stopping of electric motors. Both PLCs are wired identically, as shown:



However, despite being wired identically, the two technicians' PLC programs are quite different. Jill's program uses *retentive coil* instructions ("Set" and "Reset" coils) while Bob's uses a "seal-in" contact instruction to perform the function of latching the motor on and off:





Explain how both of these PLC programs function properly to control the starting and stopping of the electric motor.

Suggestions for Socratic discussion

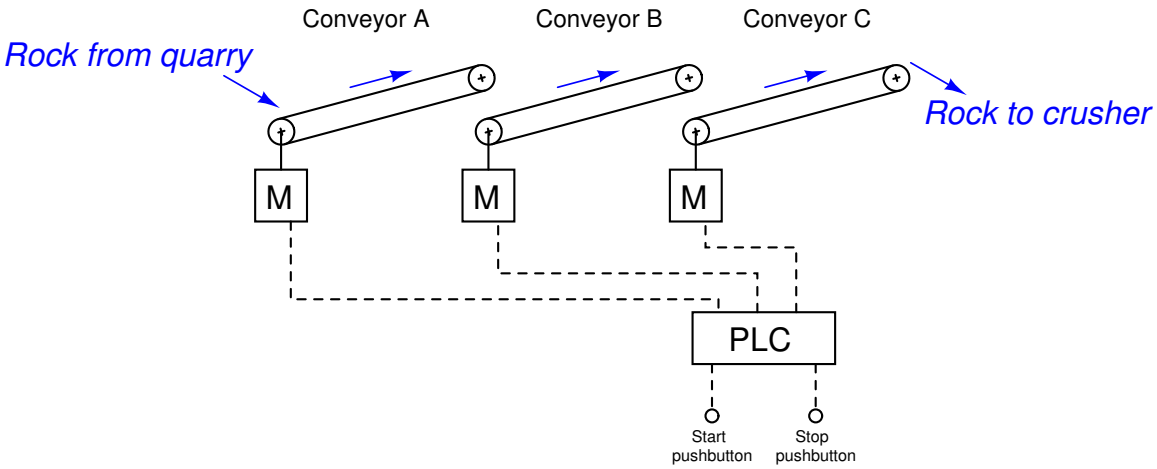
- It is ordinarily a bad thing to assign identical bit addresses to multiple coil instructions in a PLC program. With Jill's retentive coil program, however, this is not only permissible but in fact necessary for its proper operation. Explain why this is.
- A common misconception of students first learning PLC programming is to think that the type of contact instruction used in the PLC program must match the type of switch contact connected to that input (e.g. "A N.O. PLC instruction must go with a N.O. switch"). Explain why this is incorrect.

- Explain how both PLC programs will react if both the "start" and "stop" pushbuttons are simultaneously pressed.
- Alter both PLC programs to be "fail-safe" (i.e. shut the motor off) if ever the stop pushbutton switch fails circuit open.

file i03674

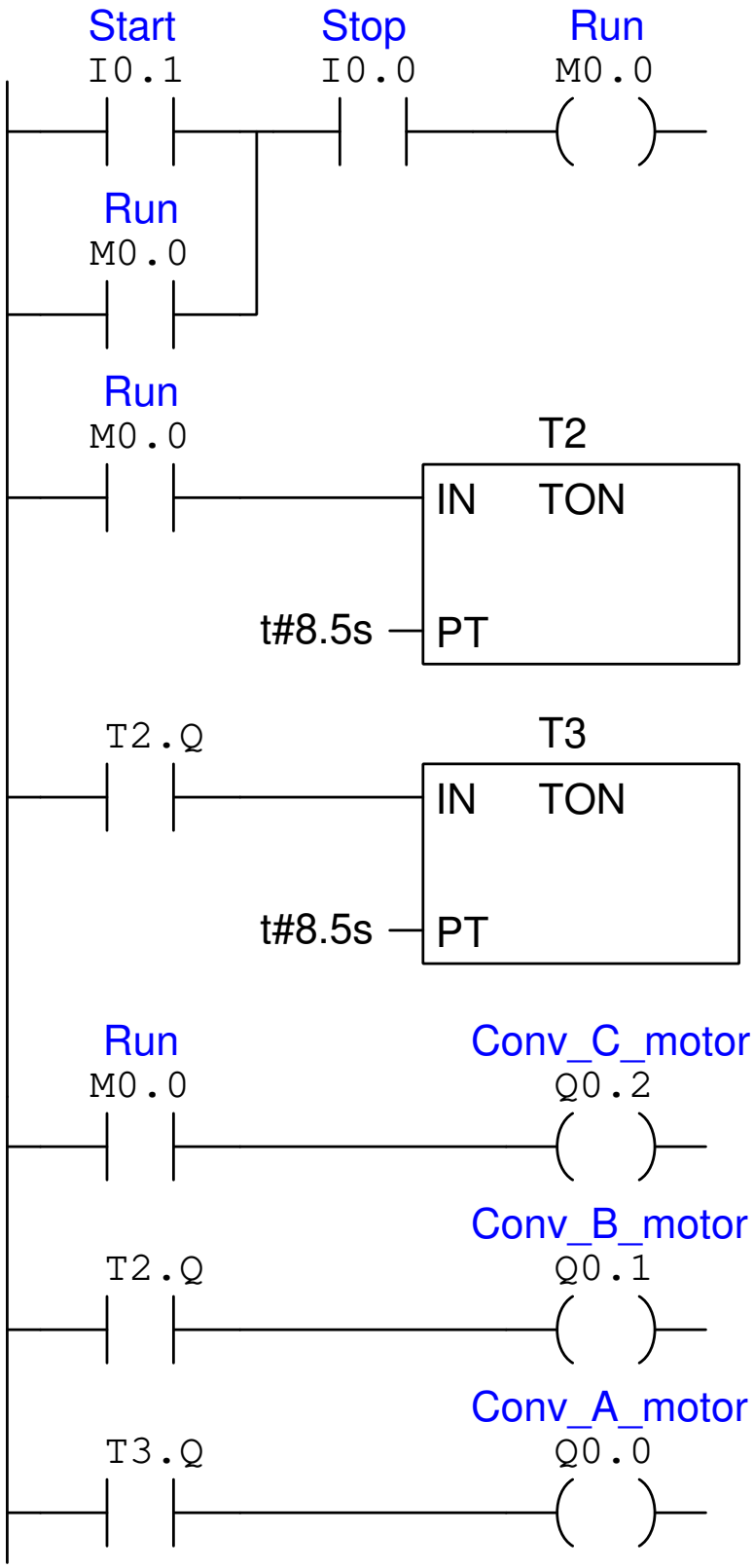
Oppgave 4

A gravel-crushing operation uses three long conveyor belts to move rock from the quarry to the crusher. The belts must be started up in a particular sequence to avoid overloading the electric motors driving them:



First, determine a start-up sequence that makes sense: which conveyor belt should start first, next, and last? What might happen if the sequence were reversed? Why not simply start all conveyor motors simultaneously?

This operation uses a Siemens S7 series PLC to control the three conveyor belts. Analyze this program and explain how it accomplishes the task of starting up the three conveyors in sequence:



Lastly, determine where you might add a contact instruction for an *emergency shutoff* safety switch, so that all three conveyors stop simultaneously if ever the safety switch is actuated.

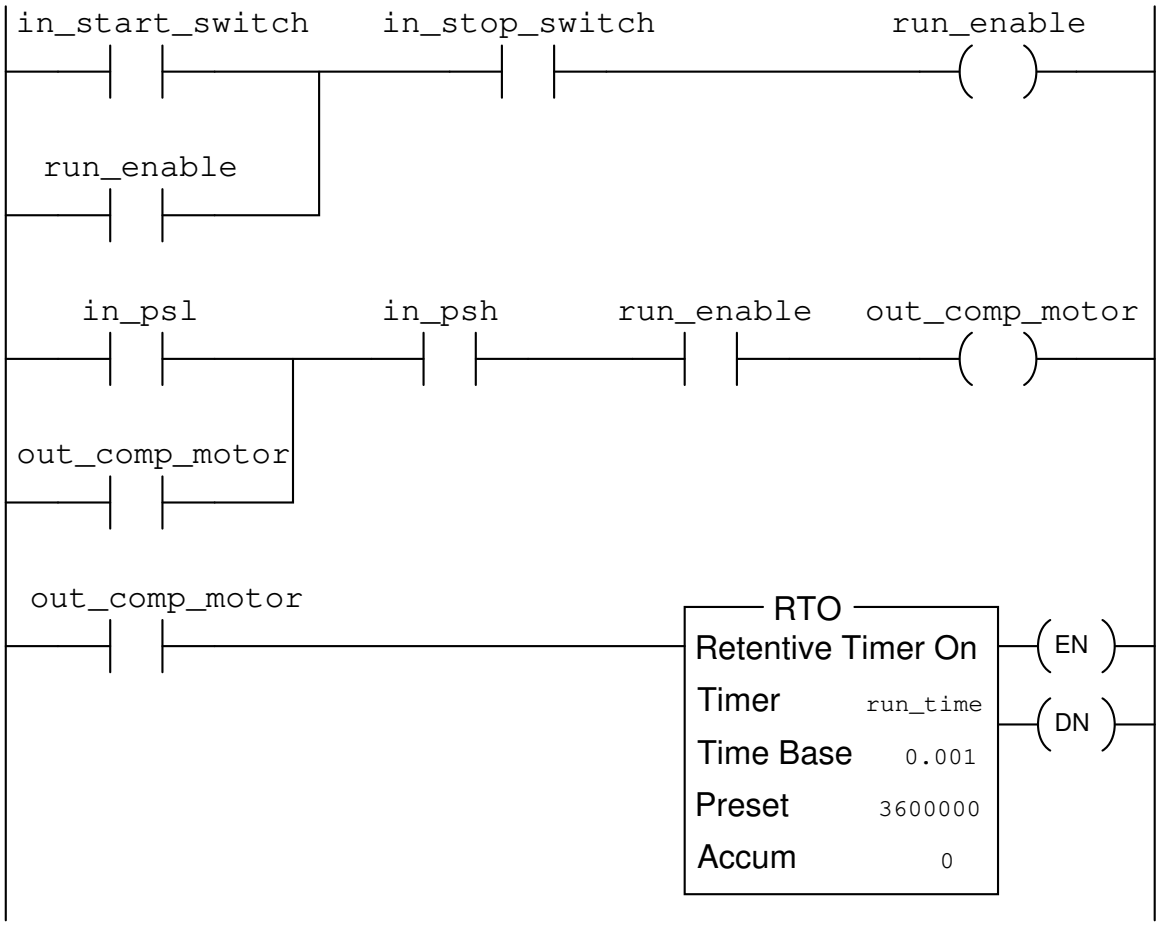
Suggestions for Socratic discussion

- How long is the time delay between conveyor start-ups? How might this time delay be altered if needed?
- Suppose a warning siren were added to the system, sounding for a full 15 seconds before the first conveyor belt starts. How would you modify the PLC program to include this additional functionality?
- Suppose a technician uses the PLC's *force* utility to force bit `T2` to a "0" state. How will this affect the operation of the system? Could the consequences of this force be dangerous in any way?
- Suppose a technician uses the PLC's *force* utility to force bit `Q0.1` to a "0" state. How will this affect the operation of the system? Could the consequences of this force be dangerous in any way?

file i04428

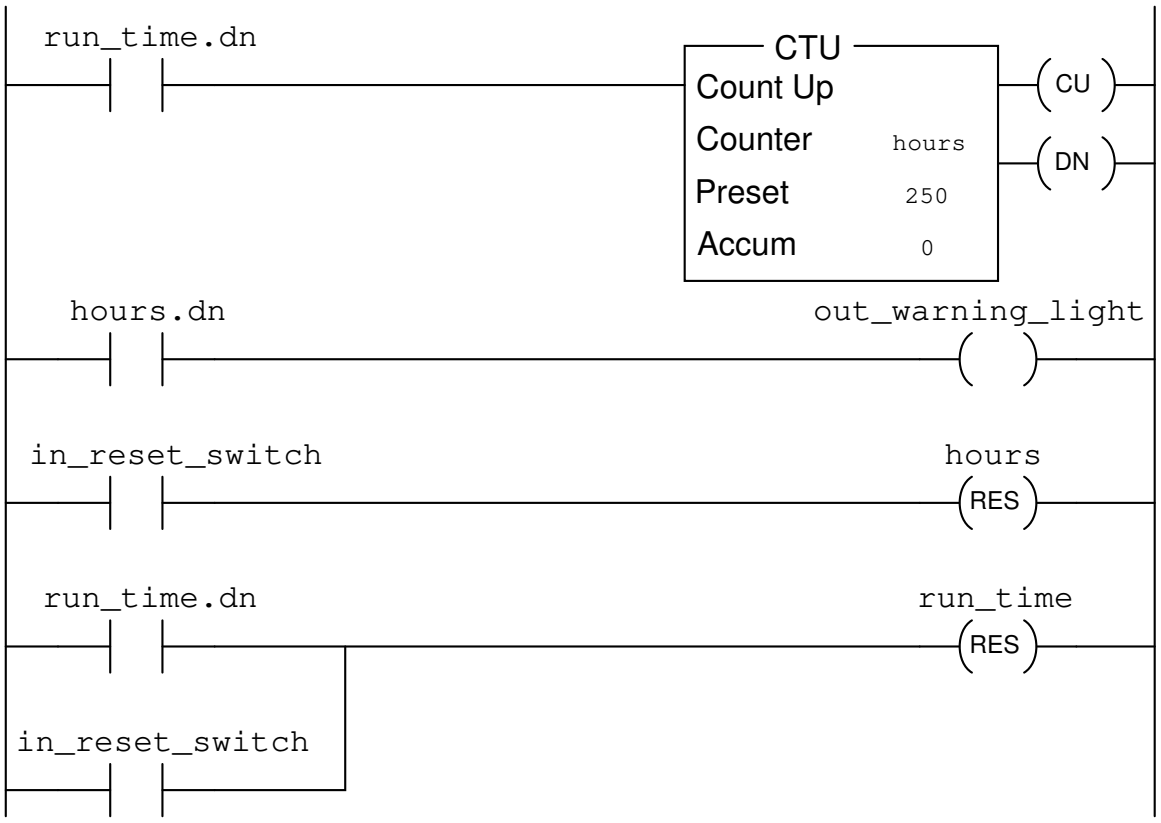
Oppgave 5

An Allen-Bradley Logix5000 PLC is used to control the starting and stopping of an air compressor based on momentary-contact pushbutton switch inputs as well as high and low pressure switches (PSH and PSL, respectively). Analyze this program and explain how it is supposed to work:



(continued on next page)

(continued from previous page)



In particular, answer these following questions:

- Determine the "normal" electrical statuses of all switches (e.g. NO or NC) connected to the inputs of this PLC, based on an examination of the respective contact instructions within the PLC program.
- Why is is important that a *retentive* timer instruction be used for the calculation of total run-time?
- What is the significance of the maintenance warning light controlled by this PLC?
- Finnes det en retentive timer til codesys

Suggestions for Socratic discussion

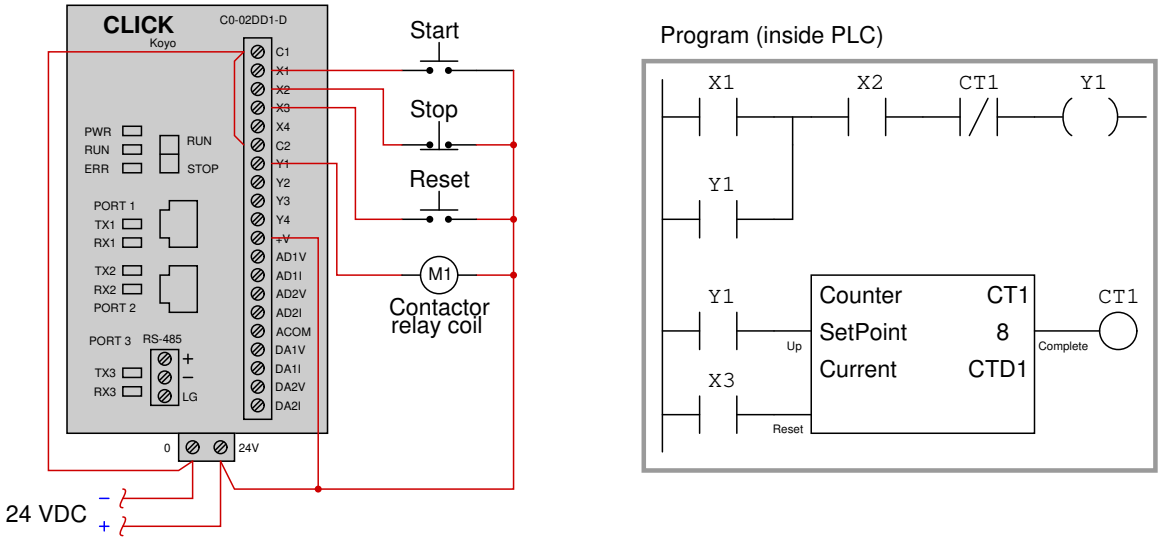
- Note how all instructions in this Logix5000 PLC program are addressed by *tagname* rather than by hardware addresses (e.g. `I:2/6`, `O:3/1`). How do you suppose the PLC "knows" which real I/O points to associate with which instructions in the program?
- How will this system behave if the reset switch fails shorted?
- How will this system behave if the high-pressure switch fails open?
- How will this system behave if the high-pressure switch fails shorted?

- How will this system behave if the low-pressure switch fails open?
- How will this system behave if the low-pressure switch fails shorted?

file i02346

Oppgave 6

This Koyo "CLICK" PLC has been programmed to control the starting and stopping of an electric motor, including a *counter* instruction to prevent the motor from being started up more than a specified number of times:



Identify the counter instruction in the program shown, its input "connections", and also how the result of the counter reaching its pre-set limit forces the motor to stop. Also, determine the maximum number of times the motor may be started up, assuming the counter's current value goes to zero when the Reset button is pressed.

Finally, determine how to modify this PLC program so that the counter may be manually reset by the operator without requiring a separate pushbutton labeled "Reset".

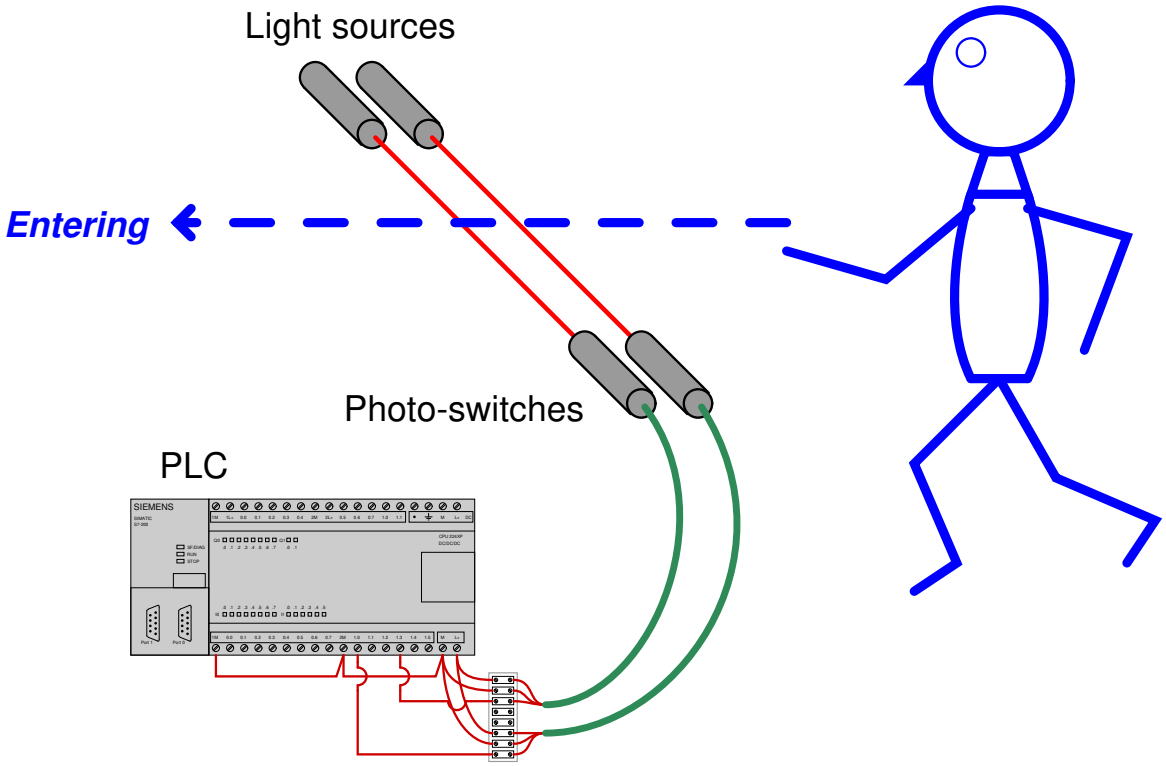
Suggestions for Socratic discussion

- If an operator presses the "Start" button multiple times while the motor is already running, do these button-presses get counted by the counter instruction, or do only the real motor start-up events get counted?
- What do you suppose the label "CTD1" represents inside the counter instruction?
- Note the number of times the bit `Y1` is referenced inside this PLC program: once in a coil instruction and twice in contact instructions. Is there any limit to how many times a bit address may be used in a PLC program?
- Describe the purpose of the first contact instruction labeled `Y1` in this program, explaining why it is often referred to as a *seal-in* contact.

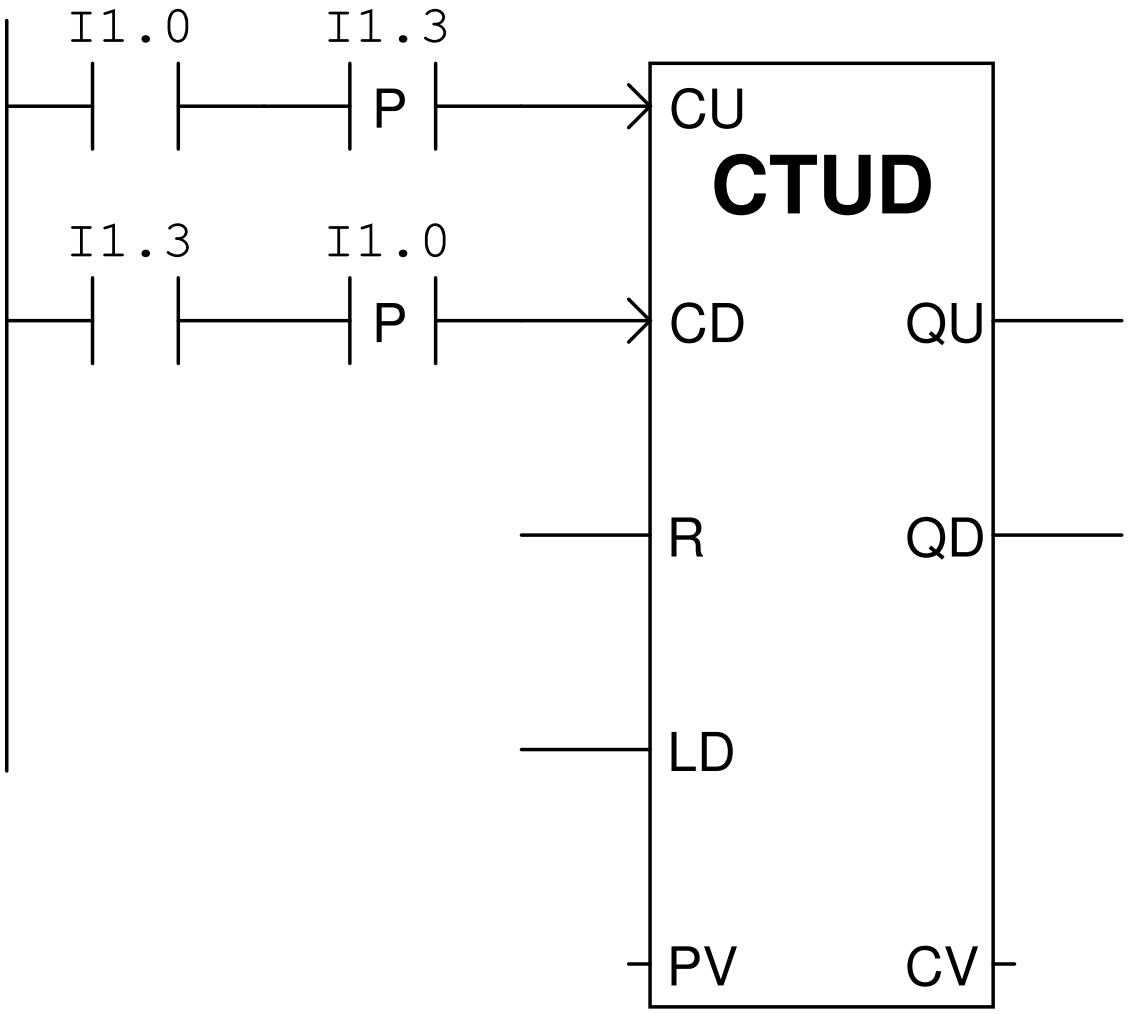
file i03589

Oppgave 7

This Siemens S7-200 PLC has been programmed to count the number of people in a room, by incrementing a counter every time a person enters through the doorway, and decrementing that same counter whenever someone exits through the same doorway. The two optical switches activate whenever their respective light beams are broken by someone passing through. Their horizontal separation is just a couple of inches – much less than the girth of a person's torso. The operating status of each switch is that it energizes the PLC input when the light beam is broken:



Examine the program in this PLC for counting people, and determine how it is able to differentiate between a person entering the room and a person leaving the room:



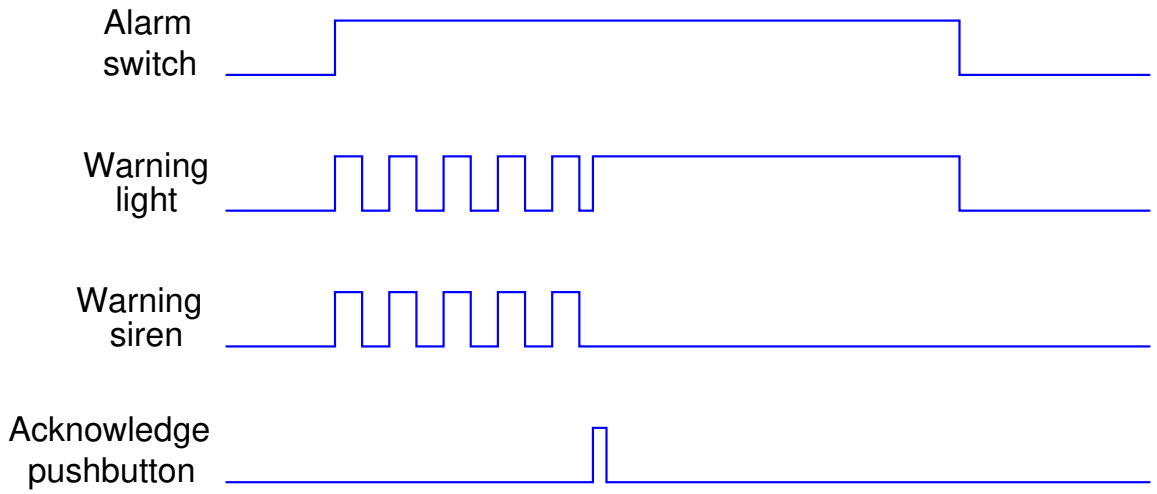
Suggestions for Socratic discussion

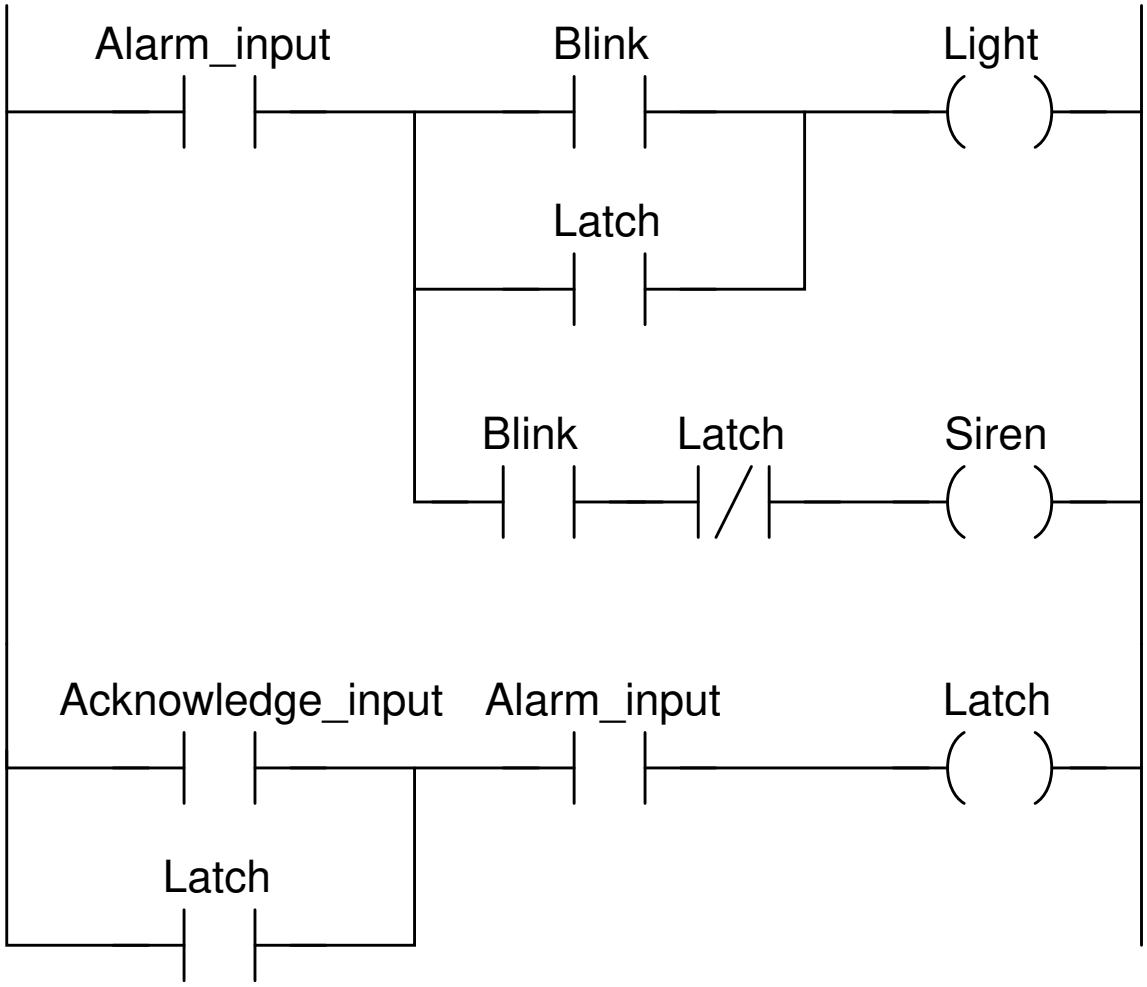
- Explain how a *timing diagram* of the switch states would be helpful in analyzing the operation of this PLC program.
- *Transition* (edge-detecting) functions are implemented in Allen-Bradley PLCs using the *one-shot rising* (OSR) instruction. Research how the OSR instruction is used, and how it differs from the "P" and "N" contacts shown in this Siemens PLC program.
- Will this system still function properly if the optical sensors are spaced farther apart than the width of a human body? Explain why or why not.

file i00185

Oppgave 8

The following PLC program preforms the function of an *alarm annunciator*, where a discrete input signal from an alarm switch (e.g. high temperature alarm) first causes a warning light to blink and a siren to audibly pulse until a human operator presses an *acknowledge* pushbutton. If the alarm switch signal is still activated, the light will remain on (steady) instead of blink and the siren will go silent. The light turns off as soon as the alarm signal goes back to its "safe" state. A timing diagram shows how this should work:





Take this "generic" PLC program and enter it into your own PLC, assigning appropriate addresses to all instructions, and demonstrating its operation.

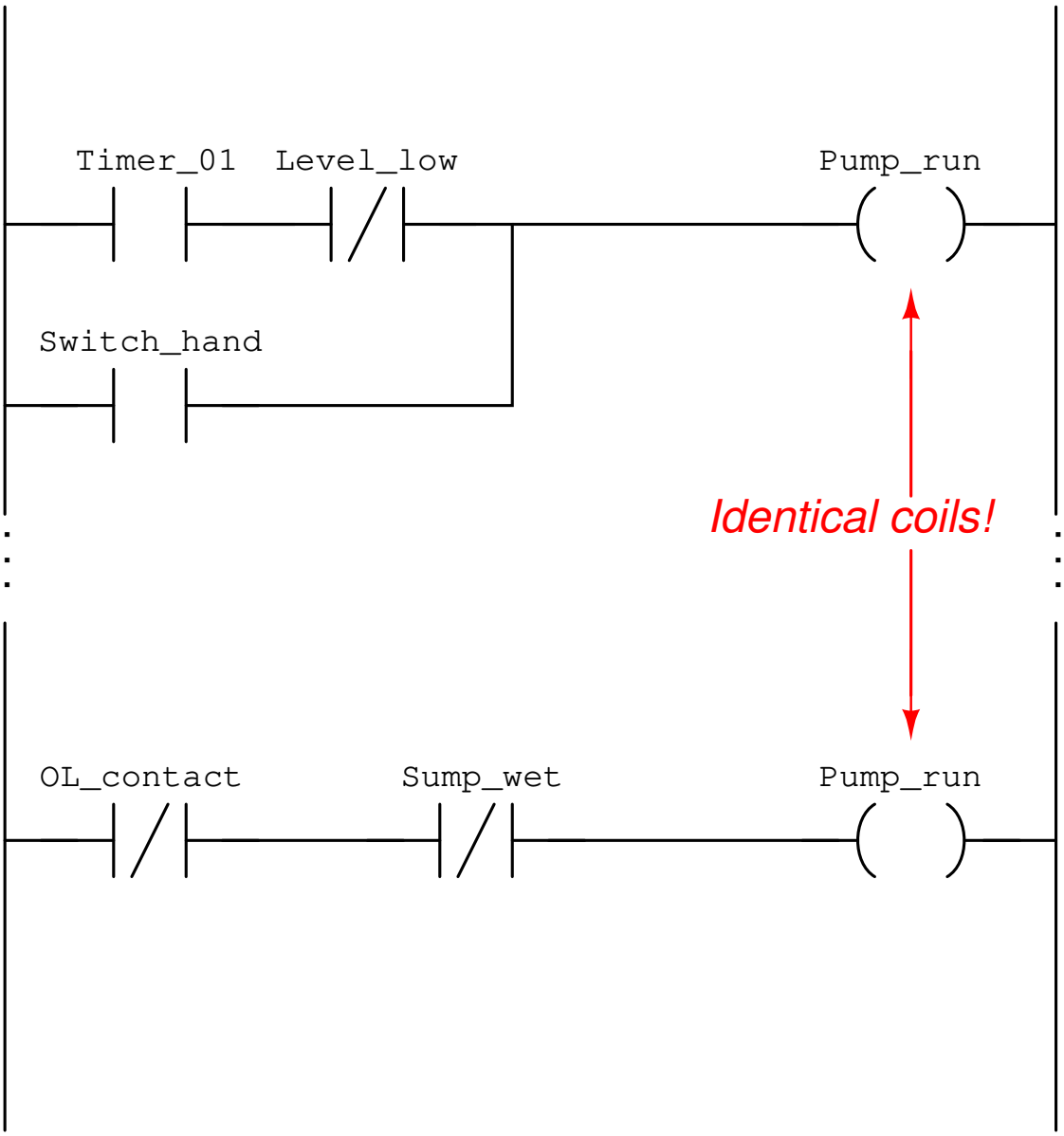
Suggestions for Socratic discussion

- Does the PLC program (as written) "expect" a *closed* alarm switch contact to trigger the alarm, or an *open* alarm switch contact?
- If the real-world alarm switch contact was a pressure switch wired NC (normally-closed), would this circuit function as a *low* pressure alarm or as a *high* pressure alarm?
- If the real-world alarm switch contact was a temperature switch wired NO (normally-open), would this circuit function as a *low* temperature alarm or as a *high* temperature alarm?

file i02342

Oppgave 9

In relay ladder logic (RLL) programming, it is considered bad practice to have multiple instances of an identical (standard) "relay" coil in a program:



Explain why this is considered poor practice in PLC programming. Next, determine the status of the `Pump_run` output channel given the following bit states:

- `Timer_01` = 1
- `Level_low` = 1
- `Switch_hand` = 0
- `OL_contact` = 0
- `Sump_wet` = 0

file i02376

Oppgave 10

Lyset i en gang opereres av to sett med impulsbrytere, et i hver ende av gangen. Disse er kobla til en PLS.

1. Lag en skisse for oppkoblingen
2. Sett opp en IO liste
3. Lag et PLS program for denne funksjonen.
4. Utvid PLS programmet til å virke med en bryter i hver enda av gangen.

file i08000

Oppgave 11

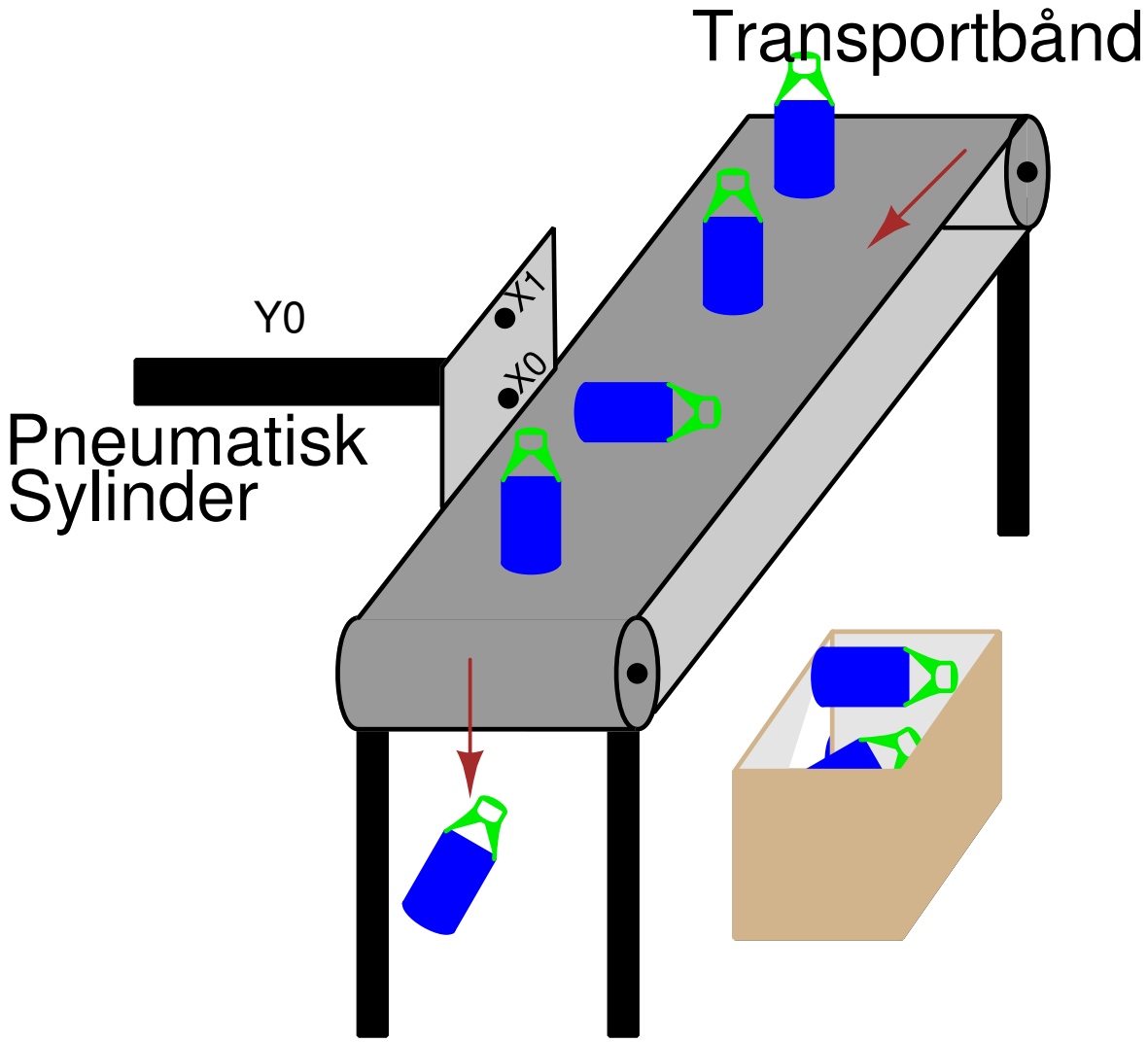
Lyset på en idrettsarena kan opereres fra to ulike innganger, A og B, med individuelle impulsbrytere som er kobla til en PLS slik som i oppgaven over. Her skal vi ha en tilleggsfunksjon: Når lyset er skrudd på ved inngang A så skal det ikke kunne slukkes ved inngang B og vice versa.

1. Lag en skisse for oppkoblingen
2. Sett opp en IO liste
3. Lag et PLS program for denne funksjonen.
4. Utvid PLS programmet til å virke med en bryter i hver enda av gangen.

file i08001

Oppgave 12

Du skal lage et program som detekterer og skubber vekk flasker som har veltet på et samlebånd. Følerene X0 og X1 har bryter av typeNC (Normaly Closed). Den pneumatiske sylinderen Y0 aktiveres med TRUE og skubber da ut sylinderen. Dette går så fort at den rekke å skubbe ut og komme tilbake før neste flaske kommer frem.



file i08002

Oppgave 13

Du skal lage styring for overvåkning av et tankanlegg med kjemikalier. Tankanlegget består av 3 separate tanker. Hver av tankene inneholder en nivåføler som gir logisk 1 når kjemikaliemengden er under et minimumsnivå kretsen sin oppgave er å tenne (gi logisk 1) en varsellampe på et kontrollpanel når kjemikaliemengden i to eller flere av tankene er under minimumsnivået.

1. Tegn en enkel skisse for å visualisere systemet for deg selv.
2. Lag et skjema for oppkoblingen
3. Sett opp en IO liste
4. Lag et PLS program for denne funksjonen.

Tilleggsoppgave. Du skal lage en alarmkrets.Den skal virke slik ved aktivering av alarm:

- En indikator skal blinke med alarm og det skal lages lyd (simuleres med lys).
- Nå det trykkes Acknowledge skal Indikatoren lys konstant og lyden skal gå av.
- Når alarm betingelsen er borte skal alarmen kunne resettes.

file i08003

Oppgave 14

Til en bil skal du lage en alarmkrets som varsler med lyssignal når sikkerhetsbeltene i forsetet ikke er festet og det sitter noen i setene. Under hvert sete er det en bryter som lukker når en person setter seg i setet. Videre er det i hvert sikkerhetsbelte en bryter som lukker når beltet blir festet.

1. Sett opp en IO liste
2. Lag et PLS program for denne funksjonen
3. Lag et HMI display som viser funksjonen

file i08004

Oppgave 15

Lyset i et rom styres av to impulsbyrtere, AV og PÅ, når av bryteren betjens skal lyset stå på i 5 sek. før det går av.

1. Sett opp en IO liste
2. Lag et PLS program for denne funksjonen.

file i08005

Oppgave 16

Foriglinger
To hydrauliske sylindre er arrangert slik at det er kollisjonsmulighet mellom dem. Derfor må operasjonen av dem være slik at for at den ene skal gå i + må den andre være i –stilling og vice versa.

Adresse	Funksjon Innganger	Adresse	Funksjon Utganger
I0	Start	Q0	Sylinder A gå i + retning
I1	Stopp/pause (når 0)	Q1	Sylinder A gå i - retning
I2	Sylinder A i -pos	Q2	Sylinder B gå i + retning
I3	Sylinder A i +pos	Q3	Sylinder B gå i - retning
I4	Sylinder B i -pos		
I5	Sylinder B i +pos		g

Lag et utgangsprogram for styringen

file i08006

Oppgave 17

På en parkeringsplass for biler er det plass til 10 biler. Det er separat inn- og utkjøring fra parkeringsplassen. Ved innkjøringen er det plassert to lamper, en som skal lyse grønt hvis det er ledige plasser, og en som skal lyse rødt når det ikke er ledige plasser. Ved innkjøring er det en bryter som gir signal for hver bil som kjører inn på plassen. Ved utkjøring er det en bryter som gir signal for hver bil som kjører ut.

1. Tegn en skisse for lysregleringen
2. Sett opp en IO liste
3. Lag et PLS program for denne funksjonen.

file i08008

Oppgave 18

Sekvensiell oppstart av tre motorer
Tre motorer skal starts når du trykker på en knapp. Oljemotoren starter umiddelbart, hovedmotoren starter etter 10s og hjelpemotoren starter etter 15s. Når en trykker stopp skal alle motorene stoppes.

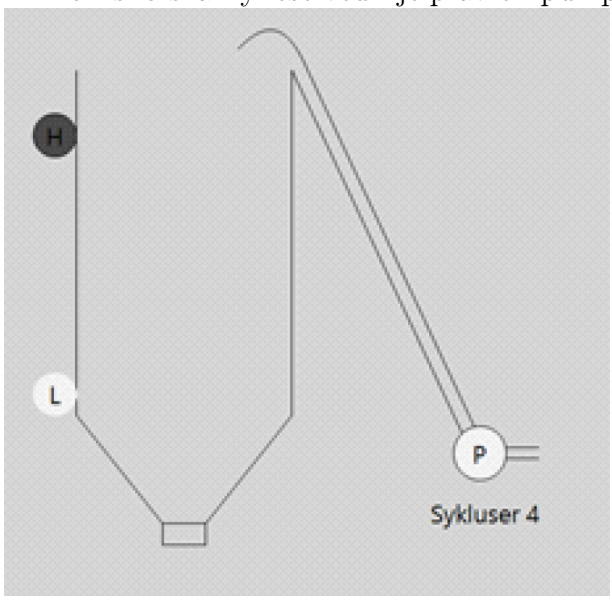
file i08009

Oppgave 19

Test oppgave Kornsilo lett versjon Simulering av kornsiloen ligger i Gand biblioteket. (SimKornsilo)

Tilkoblet utstyr	IO på RIO	Variabel	Beskrivelse av tilkoblet utstyr
Start Knapp	Bryter1	Start	
Stopp Knapp	Bryter2	Stopp	
H Sensor	Bryter3	LevelHigh	
L Sensor	Bryter4	LevelLow	
Drifts Lys	Lys1	Drift	
Pumpe Lys	Lys2	Pumpe	
Alarm Lys	Lys3	AlarmLys	
Alarm Lyd	Lys4	AlarmLyd	

En kornsilo skal fyllest ved hjelp av en pumpe



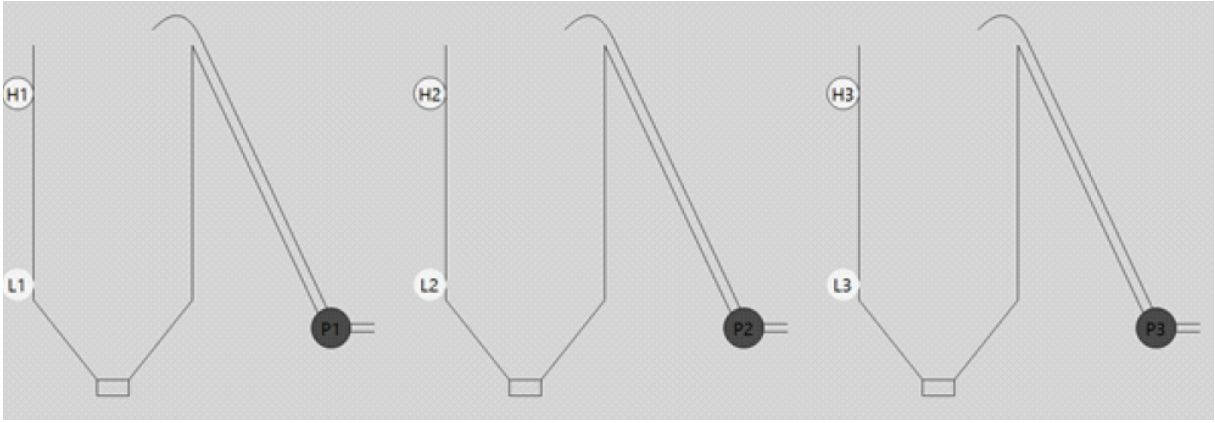
Der er to nivåvakter i hver silo (L og H) disse gir " TRUE" når nivået ligger over giveren. Virkemåte: Pumpa i en silo skal alltid starte når nivået er under minimum og stoppe når nivået går over maksimum for siloen.

1. Lag en Visualisering som ligner den på bildet
2. Anlegget settes i drift med en Start knapp og stoppes med en Stoppknapp. Når anlegget er satt i drift og nivået er under L startes pumpe P. Denne går til nivået når H. Slik fortsetter det til driften av anlegget stoppes.
3. Legg til AutoMan styring av pumpe
4. Legg til en teller for totalt antall sykluser på pumpe (skal vises på skjerm).
5. Det skal aktiveres en alarm om det tar mer en 10min (10s) å komme over L nivå (etter at den har vært under). Alarmen skal ha bekreft og resett funksjon.

Test oppgave kornsile vanskelig versjon Tre kornsiloer skal fyllest ved hjelp av hver sin pumpe P1, P2 og P3.

Tilkoblet utstyr	IO på RIO	Variabel	Beskrivelse av tilkoblet utstyr
Start Knapp	Bryter1	Start	
Stopp Knapp	Bryter2	Stopp	
H Sensor	Bryter3	LevelHigh	
L Sensor	Bryter4	LevelLow	
Drifts Lys	Lys1	Drift	
Pumpe Lys	Lys2	Pumpe	
Alarm Lys	Lys3	AlarmLys	
Alarm Lyd	Lys4	AlarmLyd	

Table 1: IO-liste for oppkobling mot Gand RIO-trainer



Der er to nivåvakter i hver silo (L1, H1, L2. osv.) og alle disse gir" 1" når nivået ligger over giveren. Pumpa i en silo skal alltid starte når nivået er under minimum og stoppe når nivået går over maksimum for siloen.
Pumpene skal styras slik at det ikke blir brukt mer enn 2000W (P1=500W, P2=1000W og P3=1500W). Ved tom tank, skal silo fyllest slik at den ikke lenger er tom (uavhengig av strømforbruk). Alle nettverk programmeres i LD/FBD. Det skal være mulighet for manuell styring av pumper.

1. Anlegget settes i drift med en Start knapp og stoppes med en Stoppknapp. Når anlegget er satt i drift og nivået er under L startes pumpe P. Denne går til nivået når H. Dette gjeler for alle pumpene. Slik fortsetter det til driften av anlegget stoppes.

2. Komplementer anlegget med alarm dersom nivået ligger under minimum i mer enn 1 minutt i en av siloene. Alarmsignalet skal pulsere (blinke) med en frekvens på 1 HZ.

3. Det er ønskelig med teller for antall sykluser og driftstimer på hver pumpe. Driftstid og antall sykluser skal presenteres ved hver pumpe. Når en pumpe kommer over 1000 sykluser eller 1000 driftstimer skal det vis et varsel om vedlikehold. Dette skal kunne resettes etter utført vedlikehold.

4. Sett opp kommunikasjon med Gand-RIO Trainer i henhold til tilordningslisten.

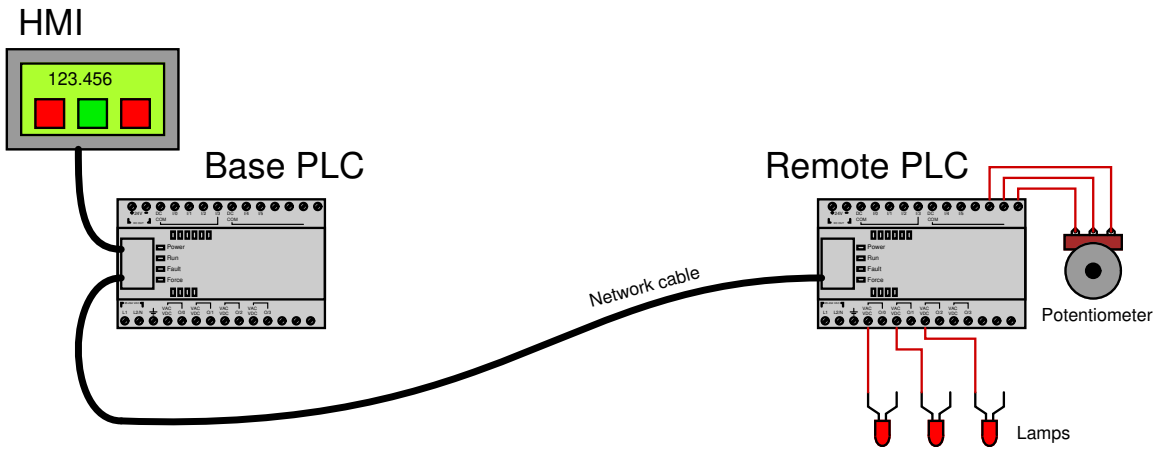
file i08012

Oppgave 20

Programming Challenge and Comparison – build a simple SCADA system

"SCADA" is an acronym meaning "Supervisory Control And Data Acquisition", referring to control systems where remote units relay data to and from a central location, allowing human operators to monitor and control processes spread over a wide area. The term "SCADA" is broadly applied to many different types of control systems in industry. Traditionally the term has been limited to control systems spread over a wide geographic area (e.g. power distribution systems, pipeline control systems) but it is now common to see "SCADA" used to describe *any* form of computer-based control system where process data is communicated over a digital network.

Work individually or in teams to wire and configure multiple PLC's to form a simple SCADA system, where analog data is read by a "remote" PLC and displayed by an HMI panel connected to a "base" PLC, and where virtual pushbuttons on the HMI display cause discrete outputs on the remote PLC to turn on and off.



When the potentiometer at the remote PLC is adjusted, the HMI at the base PLC should display a changing value. When buttons on the HMI screen are toggled, indicator lamps at the remote PLC should turn on and off. Feel free to include the following additional features for more fun and challenge:

- Build a "trend graph" display on the HMI instead of a simple numerical indicator for the analog input data point
- Have discrete inputs at the remote PLC register on the HMI display as graphic indicators
- Incorporate features such as input timers, event counts, etc. in the remote PLC
- Add multiple remote PLCs (use multi-drop RS-485 serial data communication, or Ethernet communication with a multi-port hub to connect the PLCs together)

Successful completion of this system will require the use of analog scaling instructions as well as network messaging instructions. All the usual caveats apply – *have fun!*

file i02799

Oppgave 21

Programming Challenge – HMI control of sprinkler valves

Suppose an instrument technician wishes to have a PLC-controlled sprinkler system in his yard, with an HMI panel inside his house with "pushbutton" graphics on the screen where he may conveniently activate sprinkler water solenoid valves. To begin this project, the technician connects two solenoid valves to two discrete outputs on his PLC: one valve opens up to send water to his fruit tree sprinkler nozzles while the other valve opens up to fire a jet of water at the nearby fire hydrant where his neighbor's dog likes to mark his territory.

Create a simple HMI project with two "pushbutton" icons on the screen. The first icon will directly activate the PLC output bit for the fruit tree sprinklers, and it needs to have a *toggle* action: pressing this icon once turns the bit on, and pressing it a second time turns it off. The second icon will directly activate the PLC output bit for the anti-dog water cannon, and it needs to have a *momentary* action: pressing this icon activates the water jet, and releasing it stops the water jet.
The PLC itself should have no instructions programmed in it (except perhaps for an `END` rung to avoid a processor error).

Suggestions for Socratic discussion

- What types of HMI data tags (boolean, integer, floating-point, ASCII, etc.) should be used for both these "pushbutton" objects?
- How do you specify the action (toggle, momentary, etc.) of the "pushbutton" icons on the HMI screen?
- What steps must you take to create appropriate tag names in the HMI for the PLC's data points?
- Explain why the PLC should have no program in it, or conversely, what bad things could happen if a program existed in the PLC with coils addressed to the same output bits the HMI was attempting to write to.

file i02350

Oppgave 22

Programming Challenge and Comparison – analog input scaling

Work individually or in teams to wire and configure a PLC's analog input to receive a variable voltage signal from a potentiometer, and then display that signal in three different forms on an HMI screen:

- Raw "count" value (directly read from the PLC's input register)
- Scaled 0.0% to 100.0% (as a fixed-point integer value)
- Scaled 0.000% to 100.000% (as a floating-point value)

One important point of caution is to ensure you do not "over-voltage" the input of your PLC. Some PLCs have rather limited voltage measurement ranges on their analog input terminals, and may actually suffer *irreparable damage* if you exceed the voltage limit. If this is the case (e.g. the PLC can tolerate a maximum of 10 volts to the analog input, but the only DC power supply you have for powering the potentiometer is 24 volts), you must include a fixed-value resistor in the potentiometer circuit in order to limit its full output voltage to an acceptable level.
Another note of caution is to ensure the potentiometer has a sufficient power rating to withstand the supply voltage. For example, connecting a 1 kΩ, $\frac{1}{2}$ watt potentiometer directly across a 24 VDC power supply is a recipe for smoke!

You are encouraged to consult with your instructor before powering the circuit up, to make sure the PLC's analog input will not be damaged by excessive voltage from the potentiometer. Show the sketch of your circuit as well as all relevant calculations, to prove that no ratings will be exceeded when powered.

After you've got the voltage limit and wiring figured out for the analog input, you should be able to turn the potentiometer throughout its full range and note the changing number in the PLC's input register for that analog input. Often, this is a "raw count" value based on the number of bits in the input register, proportional to the input voltage but not actually scaled in volts. Your next step will be to use math instructions in the PLC program to "scale" this raw analog input value into 0-100% to be displayed on the HMI screen.

PLC comparison:

- Allen-Bradley Logix 5000: the I/O configuration menu (specifically, the *Module Properties* window) allows you to directly and easily scale analog input signal ranges into any arbitrary numerical range desired. Floating-point ("REAL") format is standard, but integer format may be chosen for faster processing of the analog signal.
- Allen-Bradley PLC-5, SLC 500, and MicroLogix: raw analog input values are 16-bit signed integers. The `SCL` and `SCP` instructions are custom-made for scaling these raw integer ADC count values into ranges of your choosing.

- Siemens S7-200: raw analog input values are 16-bit signed integers. Interestingly, the S7-200 PLC provides built-in potentiometers assigned to special word registers (`SMB28` and `SMB29`) with an 8-bit (0-255 count) range. These values may be used for any suitable purpose, including combination with the raw analog input register values in order to provide mechanical calibration adjustments for the analog input(s).

- Koyo (Automation Direct) DirectLogic: you must use standard math instructions (e.g. `ADD`, `MUL`) to implement a $y = mx + b$ linear equation for scaling purposes.

- Koyo (Automation Direct) CLICK: the I/O configuration menu allows you to directly and easily scale analog input signal ranges into any arbitrary numerical range desired.

file i02575

Oppgave 23

Identify a few of the "function codes" specified within the *Modbus* standard used to read and write data between industrial devices, and provide the appropriate Modbus address ranges for each of the function codes you list.

file i02384

Oppgave 24

Suppose a technician needs to program a PLC to take the raw analog-to-digital "count" value from an analog input card and scale it to a value ranging 0 to 100 (%). The input card's ADC count range is 0 to 65535. The standard formula for doing this conversion is as follows:

$$\text{Scaled output} = \frac{\text{Raw input}}{65535} \times 100$$

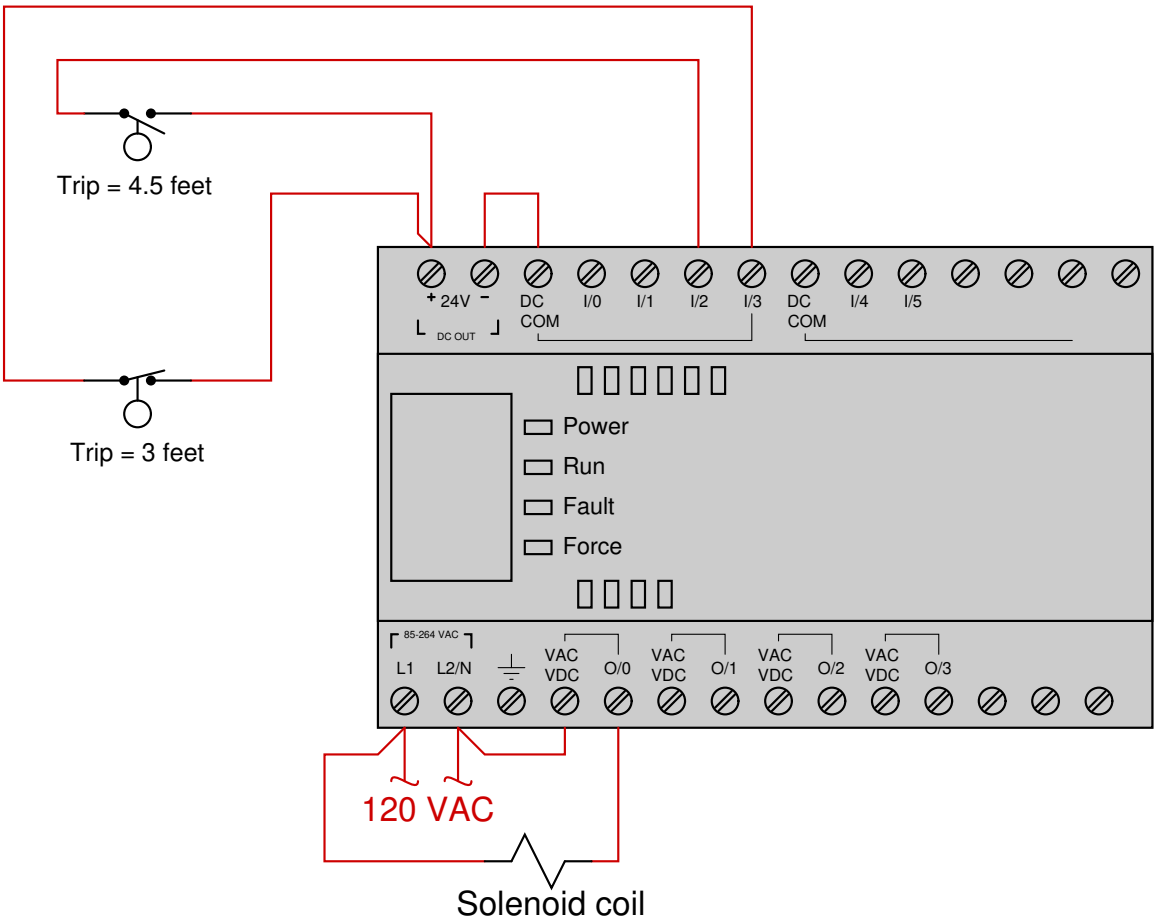
Ideally, this formula entered into a "Math" instruction in the PLC will convert any raw count value from the analog input channel into a 0 to 100% value. However, when the technician tries programming this formula into the PLC's math instruction, the result is always either 0 or 100 and never any other values. After fruitlessly trying to figure out what is going wrong, a more experienced programmer walks by to observe and comments, "That's because this PLC's math instruction only does *integer* calculations." The first technician is still perplexed, and comes to you for help.

First, explain why the formula does not compute as the technician expects it to. Second, recommend a fix so that the PLC will do a better job of scaling this ADC count value into percent.

file i02381

Oppgave 25

Suppose we have an Allen-Bradley MicroLogix 1000 PLC connected to two liquid level switches installed in the same tank, controlling a solenoid valve to empty liquid out of that tank:



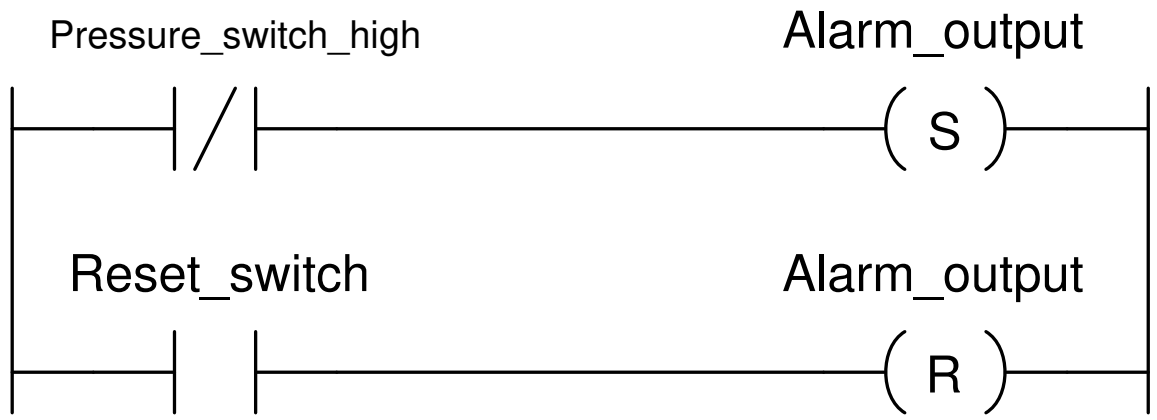
We wish for the solenoid valve to energize and open when the liquid level in the tank reaches 4.5 feet, then de-energize and shut when the liquid level falls to 3 feet. Write a RLL program for the PLC (complete with correct address labels for each of the virtual contacts) to fulfill this function:

file i02257

Oppgave 26

Programming Challenge – Alarm event latch and history timers

A normally-closed (NC) high-pressure sensing switch monitors fluid pressure in a chemical reactor vessel, opening its contacts if the pressure exceeds the trip point. This triggers an alarm lamp to energize in the control room, and this lamp will latch in the "on" state until an operator resets it, even if the high-pressure condition "clears" and goes back to normal. This is so the operators will know a high-pressure event occurred even if they were not in the control room to see it when it happened. A PLC implements this latching function using retentive ("set" and "reset") coils:



The system works well, but the operators want more. If they arrive at the control room to see the alarm light on (latched), they want to know how long the high-pressure condition lasted and also how long it's been since the reactor pressure returned to normal.

Add instructions to this PLC program to provide the desired timing functionality.

Suggestions for Socratic discussion

- Explain why the PLC program contact for the high-pressure switch is *normally-closed*, and how this information alone would be enough for us to determine that the high-pressure switch itself had NC contacts.
- What type of timer instruction is best suited for the event duration timer, a *retentive* or a *non-retentive* timer?
- How could a *counter* instruction be added to this PLC program to provide useful functionality?

file i02265

Oppgave 27

Programming Challenge – Four-function calculator

Write a PLC program and corresponding HMI project to make a simple four-function (add, subtract, multiply, and divide) calculator taking two integer values input by the user and displaying the sum, difference, product, and quotient of those two input values on the HMI screen.

Suggestions for Socratic discussion

- Determine how you could safely "experiment" with your PLC's math instructions to determine how it handles conditions such as "divide-by-zero".

PLC comparison:

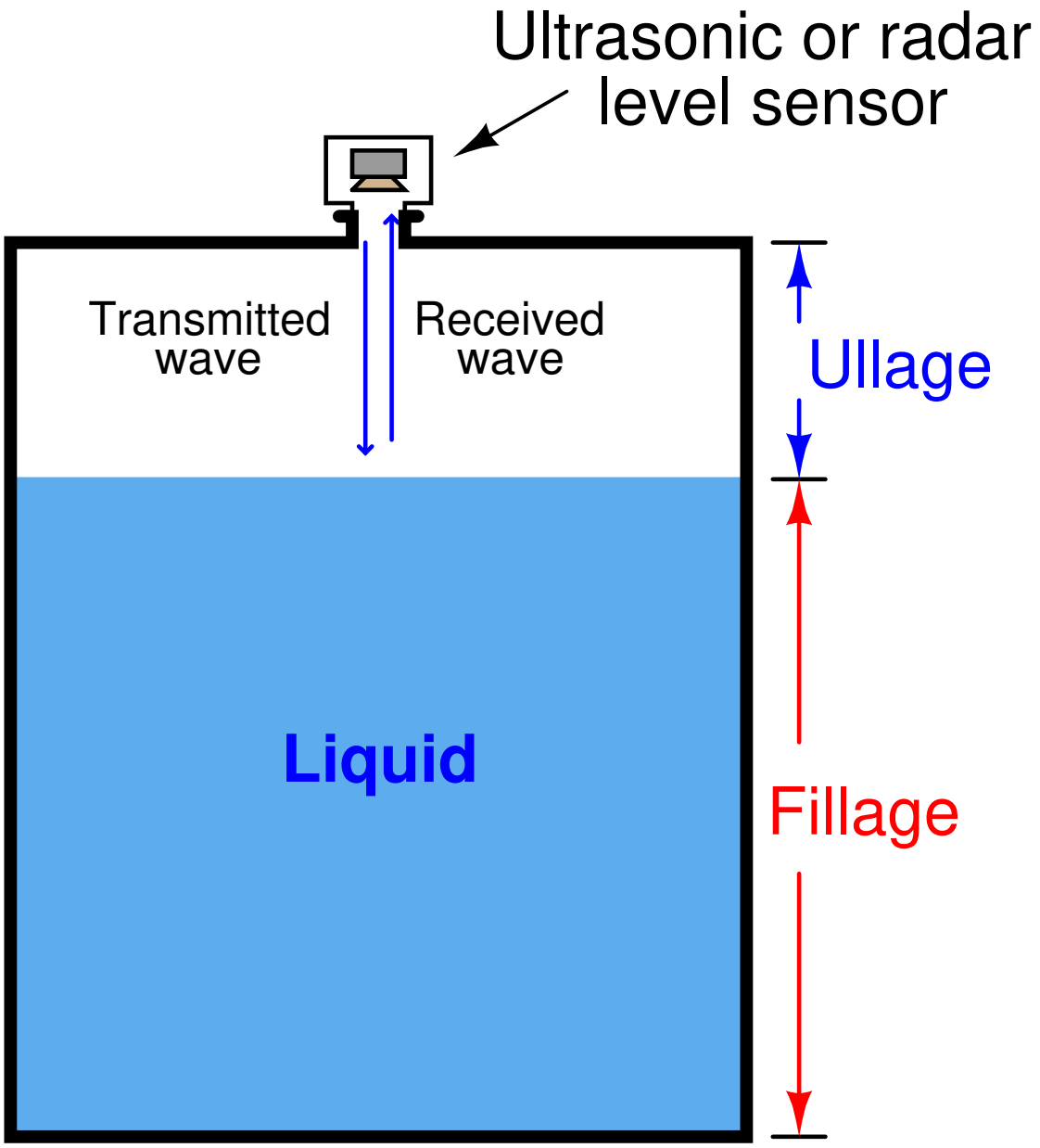
- Allen-Bradley Logix 5000: `CMP`, `ADD`, `SUB`, `MUL`, and `DIV` instructions
- Allen-Bradley SLC 500: `ADD`, `SUB`, `MUL`, and `DIV` instructions
- Siemens S7-200: `ADD_I`, `SUB_I`, `MUL_I`, and `DIV_I` instructions
- Koyo (Automation Direct) DirectLogic: `ADD`, `SUB`, `MUL`, and `DIV` instructions

file i02385

Oppgave 28

Programming Challenge – Fillage/ullage calculator

Ultrasonic- and radar-based liquid level sensing instruments where the sensor is located on the top of a storage vessel and waves are sent down to the liquid level and then reflected back naturally measure the "air space" above the liquid. The technical term for this measurement is *ullage*, representing the empty space of the storage vessel:



However, operations personnel are often more interested in the *fillage* of a vessel (how full it is) rather than its ullage. Think of it as the classic question of whether the glass is half-full or half-empty, with an industrial flavor.

Write a PLC program to take the ullage value of an ultrasonic level sensor and convert this into a fillage value for a vessel, given a fixed (total) height for the vessel. Since you probably do not have a level transmitter readily available to connect to your PLC for this exercise, feel free to simulate one by using a pair of discrete inputs to increment

and decrement an up/down counter, generating a variable simulated value for the level transmitter. If your PLC happens to have an analog input channel, feel free to input a variable voltage signal to simulate the scale's reading instead!

Suggestions for Socratic discussion

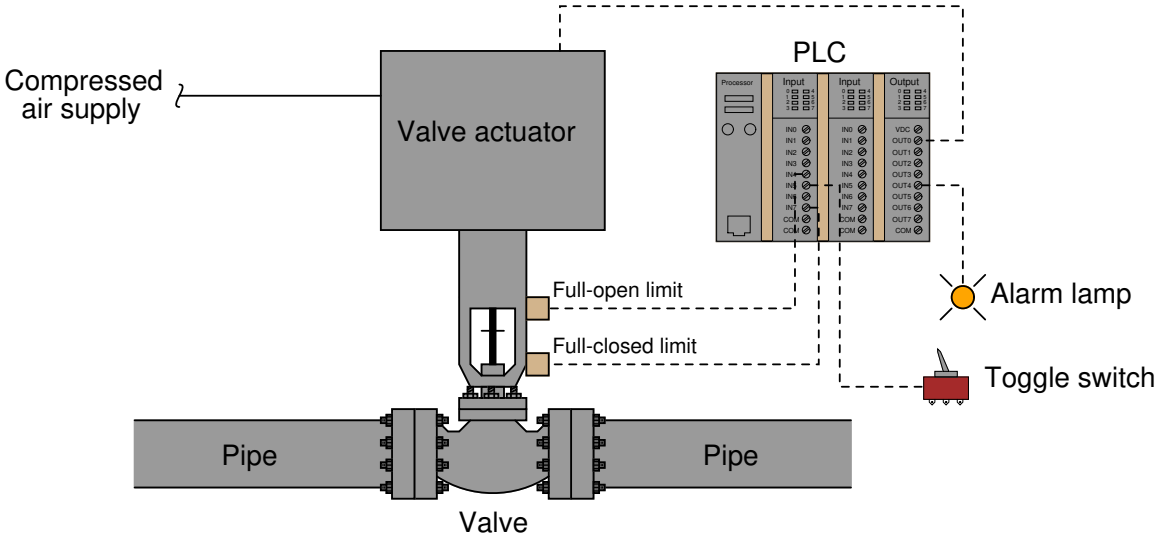
- For those who have studied level measurement technologies, what other liquid level-sensing technologies naturally sense ullage besides radar and ultrasonic?
- For those who have studied level measurement technologies, describe the difference between *guided-wave* radar level sensors and *unguided* radar level sensors.
- Determine how it is possible to format a vertical bargraph on your HMI display so that it looks like a filling tank (a very wide bargraph!), and link that bargraph's tag name to the fillage variable in your PLC.

file i02391

Oppgave 29

Programming Challenge and Comparison – solenoid valve control with stuck valve alarm

A PLC is used to control the opening and closing of a solenoid-operated valve with a single discrete output. A pair of normally-open limit switches sense the valve's stem position:



Write a PLC program energizing an alarm lamp if the valve fails to reach the full-open position within 5 seconds of receiving the "open" command signal, and energizing the same alarm lamp if the valve fails to reach the full-closed position within 8 seconds of receiving the "close" command signal. Note that the status of both limit switches will be "open" (off) when the stem is between its full-open and full-closed positions. The PLC receives the command to open or close the valve from a hand-operated toggle switch.

- **Inputs**
- Open/Close toggle – *off when commanding valve to shut ; on when commanding valve to open wide*
- Valve closed limit (NO) – *closes when valve reaches 0% position*
- Valve open limit (NO) – *closes when valve reaches 100% position*
- **Outputs**
- Valve actuator solenoid – *energizing this coil opens up the valve, de-energizing this coil allows the valve to spring-return shut*
- "Valve stuck" alarm lamp – *energize if valve does not respond in time*

When your program is complete and tested, capture a screen-shot of it as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program (either individually or as a team), prepare to discuss the following points:

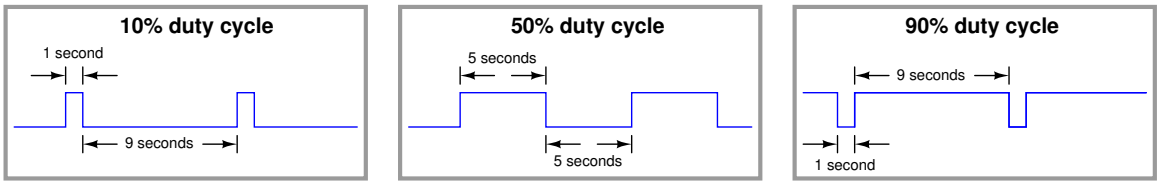
- Identify the "tag names" or "nicknames" used within your program to label I/O and other bits in memory
- Follow the sequence of operation in your program, simulating the system in action
- Identify any special or otherwise non-standard instructions used in your program, and explain why you decided to take that approach
- Show the comments placed in your program, to help explain how and why it works
- How you designed the program (i.e. what steps you took to go from a concept to a working program)

file i04657

Oppgave 30

Programming Challenge and Comparison – HMI-driven PWM duty cycle control

Suppose we wish to use a PLC to control the average amount of electrical power delivered to an oven's heating element. The simplest way to implement this control is to have the PLC output a pulsing discrete signal to a solid-state relay (SSR) which then switches AC power to the heating element, the "duty cycle" of that pulsing being adjustable between 0% and 100%, inclusive.



Write a PLC program providing this pulse-width-modulation (PWM) control, using an HMI screen to provide operators with arbitrary adjustment of the duty cycle. The frequency of this pulsing should be slow: 1 Hz or less.

When your program is complete and tested, capture a screen-shot of it as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program, prepare to discuss the following points:

- Identify the "tag names" or "nicknames" used within your program to label I/O and other bits in memory
- Follow the sequence of operation in your program, simulating the system in action
- Identify any special or otherwise non-standard instructions used in your program, and explain why you decided to take that approach
- Show the comments placed in your program, to help explain how and why it works
- How you designed the program (i.e. what steps you took to go from a concept to a working program)

Suggestions for Socratic discussion

- What type of timer instruction(s) are best suited for this application?
- Would there be an easy way to build a high limit into this system, so the operators could not increment the duty cycle value greater than 100%?
- How could you make the frequency adjustable from the HMI as well?

file i03515

Oppgave 31

Programming Challenge – HMI-driven up/down setpoint control

In an industrial process controlled by a PLC, the operators desire to have pushbutton control over the setpoint of a control loop. In other words, they want one pushbutton to increment the setpoint value of the loop whenever it is pushed, and another pushbutton to decrement the setpoint value of the loop whenever pushed. Furthermore, they want both these "pushbuttons" to be on an HMI screen rather than be real hard-wired pushbutton switches.

Write a PLC program to provide this up/down control over an integer value, and an HMI screen with the appropriate "pushbutton" icons and numerical display for the setpoint.

Suggestions for Socratic discussion

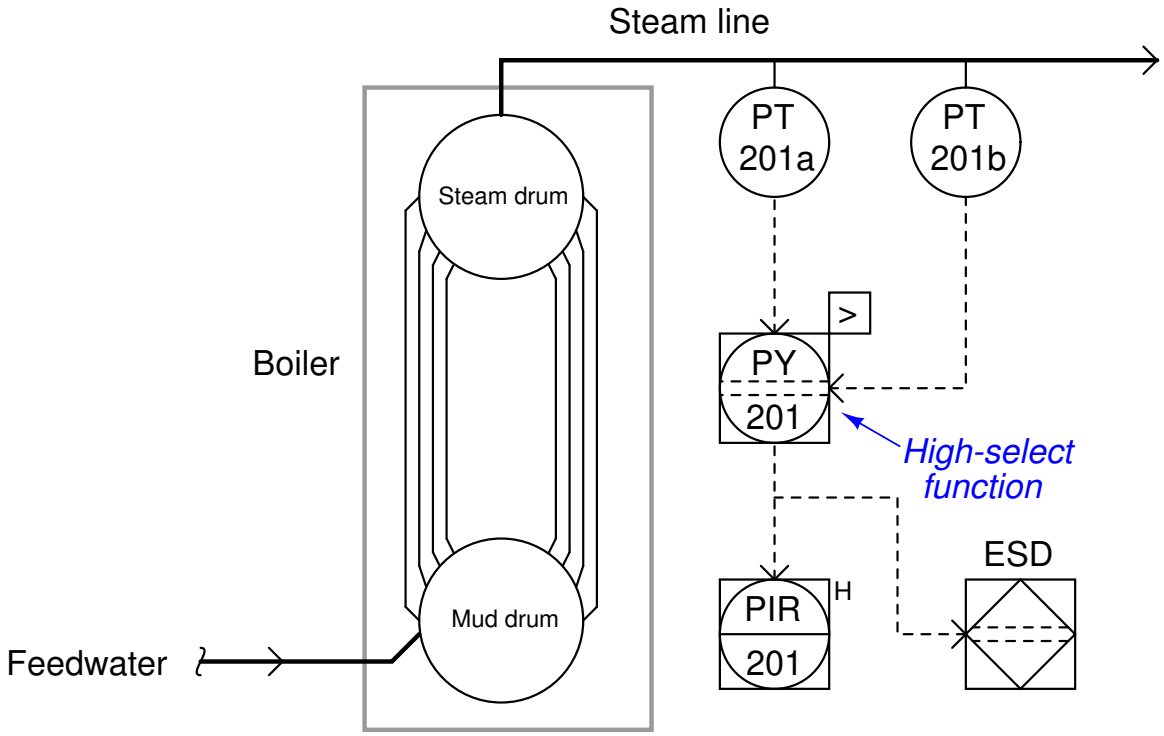
- What type of counter instruction is best suited for this application?
- Would there be an easy way to build a high limit into this system, so the operators could not increment the setpoint value any greater than a pre-set limit value?

file i02389

Oppgave 32

Programming Challenge – High-select function

A very useful type of function in instrument control systems is a *signal selector*, selecting either the highest or the lowest value among multiple input signals. These signal selector functions are particularly useful in safety control systems for "voting" between the signals of redundant transmitters.
Suppose we had an application where two pressure sensors measured the pressure of steam coming from a boiler, and we wished to know the *greatest* of these two measured pressures in case one of the transmitters were to fail with a low output signal:



Write a PLC program and corresponding HMI project for a *high-select* function, the HMI displays only one pressure readout, and the PLC selects between the greater of two input values for the HMI to read. Feel free to use a pair of up/down counters to simulate the two analog signals received by redundant transmitters (unless your PLC has two analog input channels, in which case you may input variable voltage signals to simulate the transmitters' readings!).
file i02491

Oppgave 33

Programming Challenge – model rocket launch timer with HMI screen

Suppose we wish to automate a model rocket launchpad using a PLC to time the launch of the rocket. When the "Countdown" pushbutton (momentary contact) is pressed, the PLC will begin a counting sequence to launch the rocket. After 10 seconds, a discrete output point on the PLC will activate to power the rocket engine's igniter.

Write a PLC program to perform this countdown function, and program an HMI to display the 10-second count from beginning to end in the form of a bargraph.

Suggestions for Socratic discussion

- How can you make this function latching, so that no one needs to hold the "Countdown" pushbutton the entire 10 seconds, but rather merely needs to press it once and release?

file i02349

Oppgave 34

Programming Challenge – Parking garage counter

Suppose we wish to count the number of cars inside a parking garage at any given time, by incrementing a counter each time a car enters the garage through the entry lane, and decrementing the same counter each time a car leaves the garage through the exit lane. One discrete input of the PLC will connect to a switch detecting the passing of each car through the garage entry, and another discrete input of the PLC will connect to a switch detecting cars passing out the garage exit. The PLC must be equipped with a way to for the garage attendant to manually reset the counter to zero.
Write a PLC program to perform this function, and demonstrate its operation using switches connected to its inputs to simulate the discrete inputs in a real application.

Suggestions for Socratic discussion

- What type of switches would you recommend to detect cars driving into the parking garage?
- How are you able to view the counter instruction's current count value as the program runs?
- Is there any way to "fool" this system so that it does not hold an accurate count of cars inside the garage?

PLC comparison:

- Allen-Bradley Logix 5000: `CTUD` count-up/down instruction
- Allen-Bradley SLC 500: `CTU` and `CTD` instructions.
- Siemens S7-200: `CTUD` count-up/down instruction
- Koyo (Automation Direct) DirectLogic: `UDC` counter instruction

file i03684

Oppgave 35

Programming Challenge and Comparison – Positive displacement flowmeter rate

A common design of flowmeter for residential water flow measurement is the *positive displacement* design, where the movement of water volume through the meter causes a mechanism to rotate, passing a known and fixed quantity of water volume through the meter for each revolution. The rotation of the flowmeter mechanism may be electrically transmitted by a magnetic reed switch actuated by a magnet on the flowmeter mechanism's rotating shaft. Actuating (closed and opened) one cycle per revolution, the reed switch produces a pulse signal representing a known and fixed measurement of water volume per switch "pulse."

Write a PLC program continuously calculating the flow rate of water through such a meter, given a meter factor of 1 gallon per switch pulse. The calculated flowrate needs to be displayed on an HMI, units of "GPM" (gallons per minute).

When your program is complete and tested, capture a screen-shot of it as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program (either individually or as a team), prepare to discuss the following points:

- Identify the "tag names" or "nicknames" used within your program to label I/O and other bits in memory
- Follow the sequence of operation in your program, simulating the system in action
- Identify any special or otherwise non-standard instructions used in your program, and explain why you decided to take that approach
- Show the comments placed in your program, to help explain how and why it works
- How you designed the program (i.e. what steps you took to go from a concept to a working program)

Suggestions for Socratic discussion

- How can you write your program to update the flow calculation more often than once per minute?
- One way to calculate flow is to count the number of gallons passed in one minute of time. Is there a way to calculate inversely: determining flow rate by measuring the amount of time elapsed between pulses?

file i02486

Oppgave 36

Programming Challenge – Reaction time measurement

Program your PLC to measure a person's reaction time in flipping a switch. The PLC should energize a light (or simply one of the discrete output indicating LEDs) telling the user when to flip an input switch, and then the PLC will measure how long it takes for the person to react to the light and flip the switch.

Suggestions for Socratic discussion

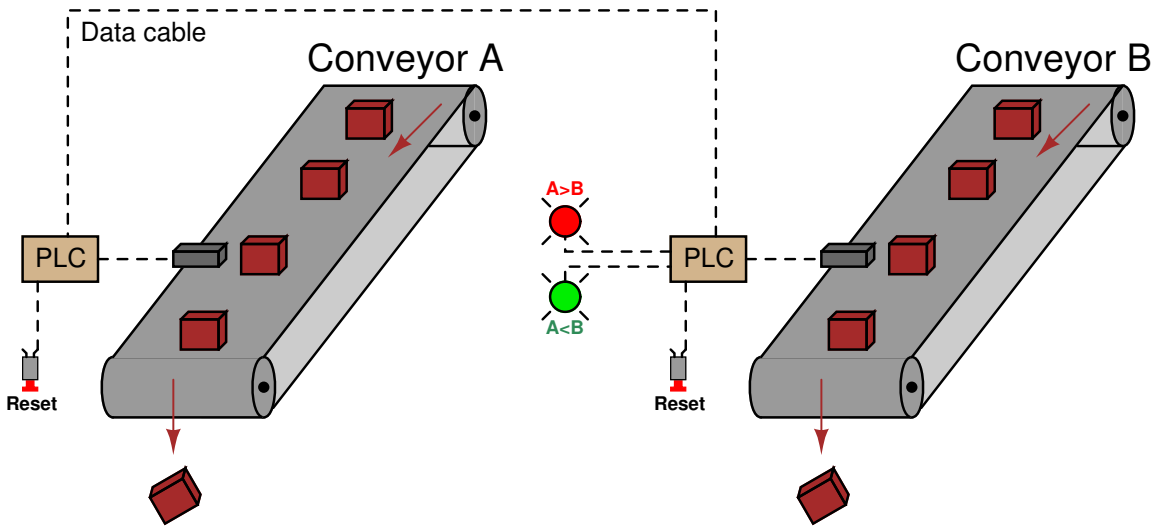
- How can you program the PLC to turn on the signaling light in a way that the person being tested cannot anticipate it?
- How must you configure the reaction time timer to count in units appropriate for this very quick time delay?
- What type of timer instruction is best suited for the reaction time timer, a *retentive* or a *non-retentive* timer?

file i02266

Oppgave 37

Programming Challenge and Comparison – Remote object counting/comparison

Suppose we have an application where two PLCs are connected via a network cable. Both PLCs count objects passing by on two separate conveyor belts using their own proximity switches. One of the PLCs needs to energize one of two lamps depending on which conveyor belt passes the most objects:



Work individually or in teams to write a PLC program comparing the two conveyor belts' parts counts using network communication instructions. Each PLC needs to have its own dedicated "reset" switch to reset that conveyor's part counter individually. *Note: some PLC models do not support the network communication ability required in this programming challenge. The Allen-Bradley MicroLogix 1000 series A and B PLCs fall into this category, being able to only respond to message queries from other PLCs, not initiate their own queries of other PLCs.*

When your system is complete and tested, capture a screen-shot of the PLC program as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program (either individually or as a team), prepare to discuss the following points:

- Show how the communication command(s) is set up, including all the relevant parameters such as baud rate, parity bits, stop bits (which must be set identically in the PLC and the other device).
- Identify which Modbus codes were used to read and/or write information with the other device.
- If multiple communication instructions were used in the PLC program, show how you programmed the PLC so these instructions would not interfere with each other (because they are each using the same communications port on the PLC).

- How you designed the program (i.e. what steps you took to go from a concept to a working program)

Suggestions for Socratic discussion

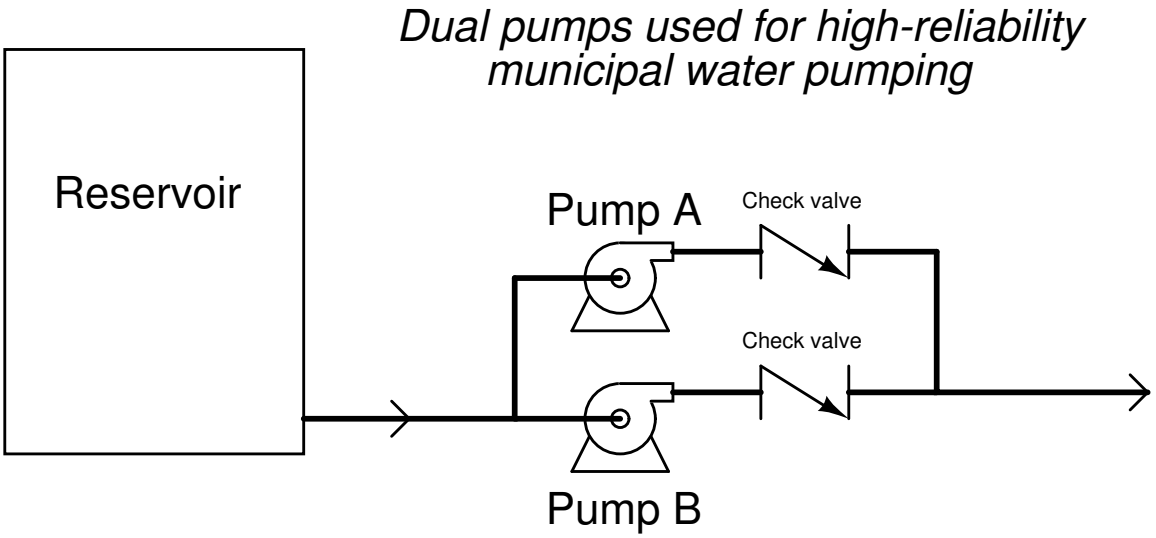
- Would you recommend one PLC in this system execute two counter functions (one for each conveyor), or would you recommend each PLC does its own counting? Explain your reasoning!
- If you program your system for duplex communication (i.e. the "master" PLC both reading from and writing to the "slave" PLC), how do you coordinate the communication instructions so that the "read" and "write" instructions happen at different times and never simultaneously?

file i02493

Oppgave 38

Programming Challenge – Run-time equalizing pump selection control

In critical process applications, it is common to find two or three pumps where a single pump would be sufficient for normal operation. Municipal water distribution and wastewater collection systems often use dual pumps for redundancy: one pump can take over for the other in the event of pump failure:



A potential problem with dual pumps is that the "spare" pump may suffer mechanical problems if it sits idle too long, and therefore will fail to perform its function as a "backup" unit should the primary pump fail for any reason. One solution to this problem is to choose the next pump to start based on which one has the least amount of accumulated run-time hours on it. Each time a pump starts, the pump to start is the one with the shortest run-time value.

Write a PLC program to take "Start" and "Stop" pushbutton switch inputs and control two pumps in this fashion.

file i00126

Svar

Svar 1

This is a graded question – no answers or hints given!

Svar 2

This is a graded question – no answers or hints given!

Svar 3

Svar 4

Although starting all three conveyor motors simultaneously would be very simple, it would be a bad thing to do because of the *inrush current* of all three motors placing undue load on the power system.

Svar 5

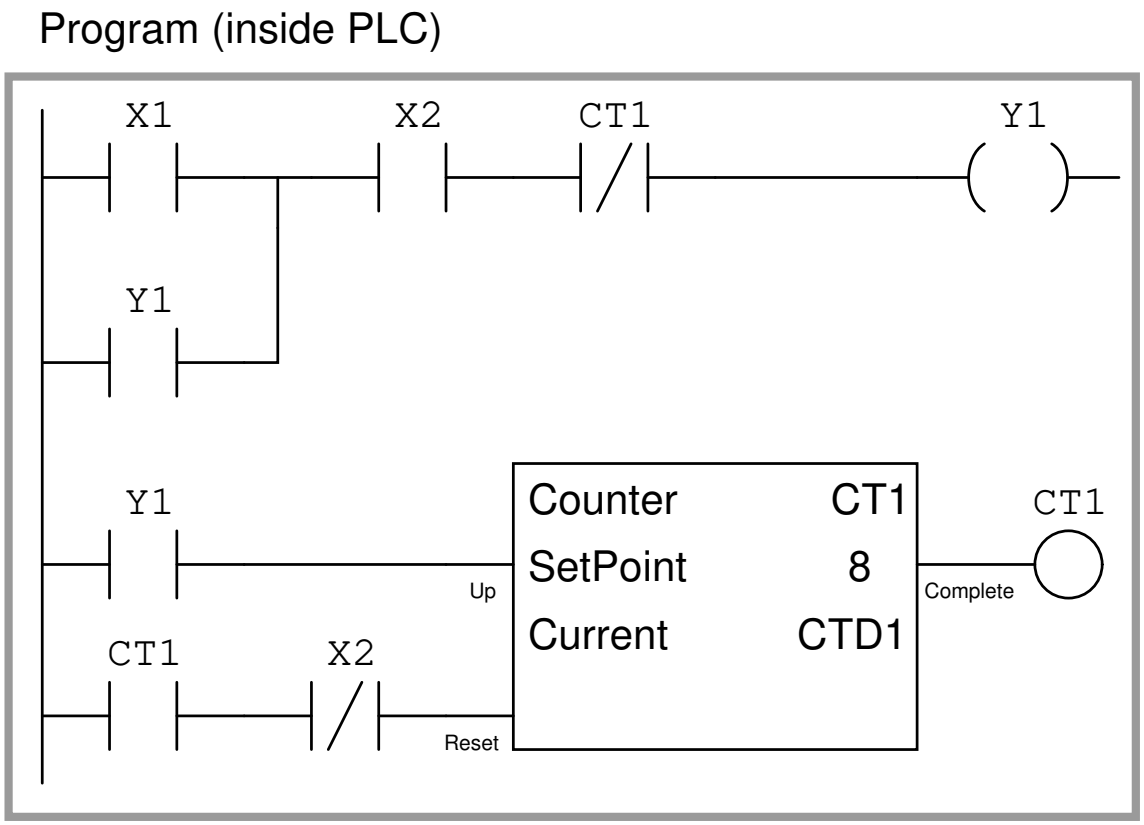
Input switch electrical "normal" statuses:

- **Start** = NO
- **Stop** = NC
- **PSL** = NC
- **PSH** = NC
- **Reset** = NO

Svar 6

This PLC program allows the motor to start up 7 times. If you thought the correct number of start-ups was eight, consider the fact that the counter's output bit (`CT1`) gets set when the counter's current value *equals* the SetPoint value, not when it *exceeds* the SetPoint value.

Here is a solution for an alternative Reset function:



In order to reset the counter, the operator must press the Stop button (after the counter has disabled the system from starting).

Svar 7

Hint: the "P" contact instructions are *positive transition* instructions, "activating" whenever their respective bits transition from 0 to 1, but returning to an "inactive" state whenever the bit value holds at either 0 or 1.

Svar 8

Svar 9

This is a graded question – no answers or hints given!

Svar 10

Det er ikke svar på denn oppgave

Svar 11

Det er ikke svar på denn oppgaven

Svar 12

Det er ikke svar på denn oppgave

Svar 13

Det er ikke svar på denn oppgave

Svar 14

Det er ikke svar på denn oppgave

Svar 15

Det er ikke svar på denne oppgave

Svar 16

Det er ikke svar på denne oppgave

Svar 17

Det er ikke svar på denne oppgave

Svar 18

Det er ikke svar på denne oppgave

Svar 19

Det er ikke svar på denne oppgave

Svar 20

Svar 21

Svar 22

A helpful reference for you, especially if programming an Allen-Bradley SLC 500 or MicroLogix PLC, is the subsection entitled "Example Calculation: PLC analog input scaling" in the "Relating 4 to 20 mA Signals to Instrument Variables" section of the "Analog Electronic Instrumentation" chapter of the *Lessons In Industrial Instrumentation* textbook.

Svar 23

This is a graded question – no answers or hints given!

Svar 24

This is a graded question – no answers or hints given!

Svar 25

This is a graded question – no answers or hints given!

Svar 26

Svar 27

Svar 28

Svar 29

Svar 30

Svar 31

Svar 32

Svar 33

Svar 34

Svar 35

Svar 36

Svar 37

Svar 38