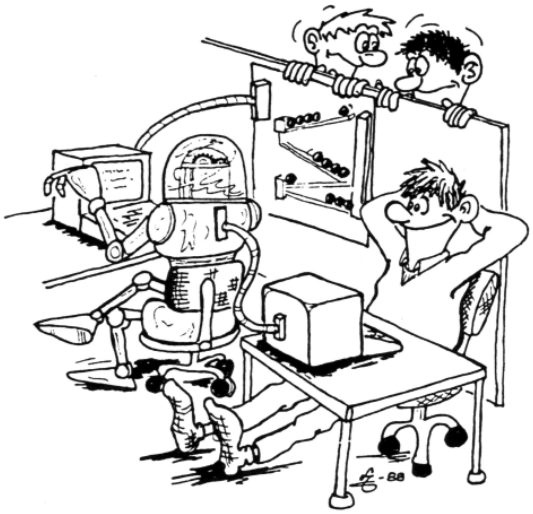
Industriautomation

Tentamen SSY 066, fredag 20/12, 08:30-10:30, V
Lärare: Kristofer Bengtsson, 0768 979561
Tid för lärarens närvaro: 9:30



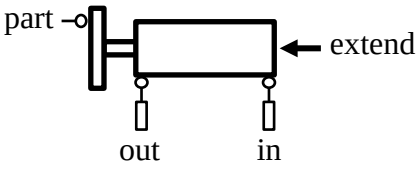
Fullständig lösning ska lämnas på samtliga uppgifter. I förekommande fall av tvetydigt formulerade tentamensuppgifter ska den föreslagna lösningen och eventuella antaganden motiveras. Examinator förbehåller sig rätten att godkänna rimligheten i antaganden och motiveringar.

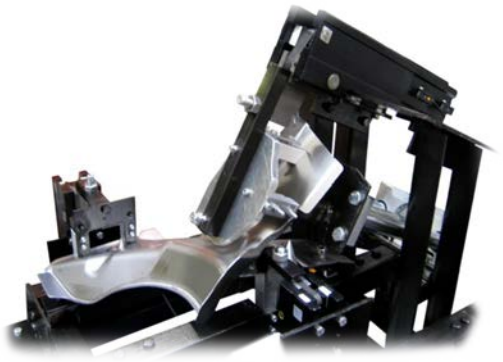
Totalt omfattar tentamen 15 poäng. För betygen tre, fyra och fem krävs 7, 10 resp 13 poäng. Lösningar anslås första vardagen efter tentatillfället på kursens hemsida i Pingpong. Granskning av rättningen sker 20 Jan kl. 12:30 – 13:30 och 21 Jan kl. 12.30 -13.30 på institutionen.

OBS. Inga hjälpmedel är tillåtna.

Uppgift 1

I denna uppgift skall vi styra en cylinder i fixturen till höger.
En förenklad bild av cylindern är:





Cylindern har två lägesgivare, insignalerna ***in*** och ***out*** som säger om cylindern är inne (***in***=true) eller ute (***out***=true). Den styrs av utsignalen ***extend***. Så länge ***extend*** är true kommer cylindern att röra sig ut och stanna ute. För att föra tillbaka cylindern skall ***extend*** sättas till false.

Det finns också en sensor ***part*** som är true om en plåtbit finns på cylinderns platta.

a) Implementera EN ladder rung som styr utsignalen ***extend***. Den skall aktiveras då er kod får styrsignalen ***move_out*** från ett annat styrsystem. Så länge ***move_out*** är true så skall ***extend*** vara true. Om ***move_out*** ändras till false skall också ***extend*** bli false förutom om det finns en plåtbit som aktiverar sensorn ***part***. Om detta är fallet skall cylindern inte åka in, förrän plåtbiten försvinner.

(2 poäng)

b) En utmaning när vi styr cylindrar är att hantera om något blir fel. I denna uppgift skall ni implementera ett larm som varnar om cylindern tar för lång tid på sig att komma helt ut. Alltså tiden det tar innan sensorn ***out*** blir true.

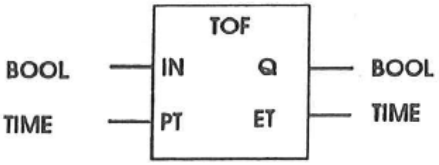
Om det tar mer än 2 sekunder (T#2s) från utsignal tills cylindern är ute, så skall signalen ***alarm*** sättas till true. Om ***alarm*** blir true så skall den fortsättningsvis vara true. Så er kod skall alltså inte återställa den. Koden skall vara skriven i ladder logic.

Olika timerfunktionsblock är beskrivna på nästa sida.

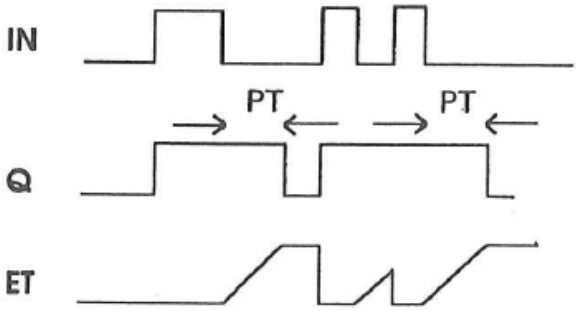
(2 poäng)

Timers

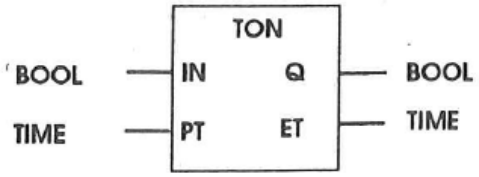
Off Timer function block



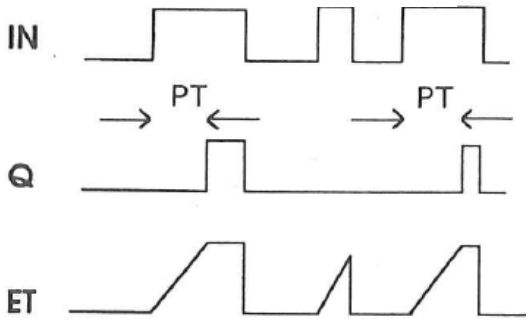
Off Timer timing diagram



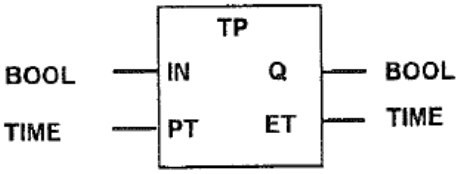
On Timer function block



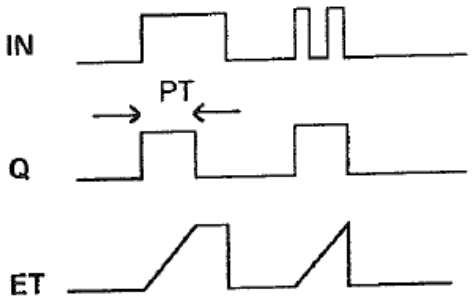
On Timer timing diagram



Puls Timer function block

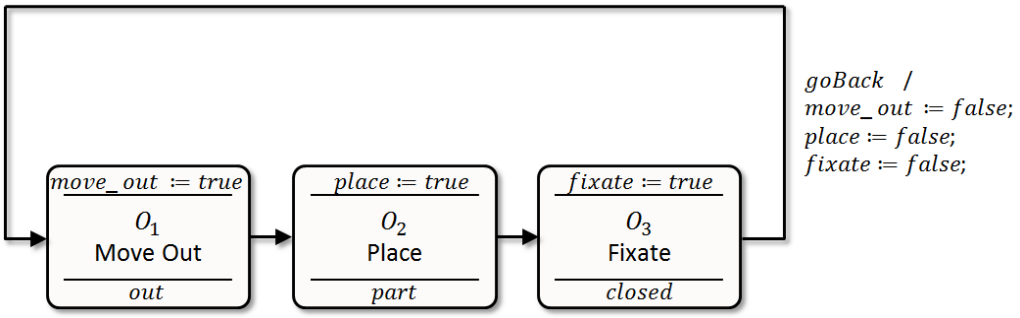


Puls Timer timing diagram



Uppgift 2

Vi fortsätter vår utveckling av fixturens styrning. Cylindern som ni programmerade i förra uppgiften används i en operationssekvens där plåtbiten fixeras och borras. Först åker cylindern ut med sin platta, sedan läggs plåtbiten på plats. Därefter fixeras den. Detta kan beskrivas med följande SOP:



Varje operation aktiverar en utsignal som en action när den startar, till exempel Place: ***place***:=true. Operationen Place är färdig när insignalen ***part*** blir sann.

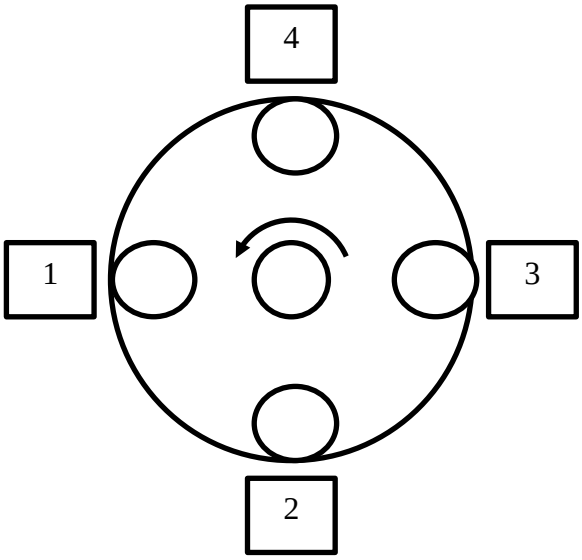
Efter att sista operationen är färdig och plåtbiten har blivit fixerad så väntar SOPen på insignalen ***goBack*** för att återställa de tre operationerna och utsignalerna. När den kommer, då återställs utsignalerna och operationerna återgår tillbaka till initialtillståndet, där sekvensen direkt kommer att upprepas. Återställningen tar inte hänsyn till nuvarande värde på insignalerna ***out***, ***part*** och ***closed.***

Er uppgift är att implementera detta beteende med hjälp av ladder logic. Observera att er kod måste befinna sig i varje operations exekveringstillstånd minst en scan-cykel även om dess postcondition är uppfyllt.

(5 poäng)

Uppgift 3

I sista uppgiften skall vi utveckla delar av styrning för att rotera ett arbetsbord. Bordet kan ni se till höger. Den har 4 arbetspositioner dit produkter kan transporteras. Den styrs med hjälp av utsignalen ***rotera***, som roterar bordet motsols när den är hög och står still när den är låg. Roteringen stannar alltså inte automatiskt i ett arbetsläge. När bordet befinner sig i arbetsposition, alltså när cirklarna befinner sig framför arbetspositionerna som på bilden, är lägesgivaren ***iLäge*** lika med true



Er uppgift är att givet en startposition via insignalen ***startPos*** = {1,2,3,4} och en slutposition via insignalen ***slutPos*** = {1,2,3,4}, rotera bordet så att produkten på startpositionen, hamnar i slutpositionen.

Till exempel: Om ***startPos*** = 1 och ***slutPos*** = 3, så skall er kod rotera bordet två steg (ett halvt varv). Om ***startPos*** = 2 och ***slutPos*** = 1 skall ni rotera tre steg. Om de har samma värde skall bordet rotera 4 steg.

Er kod skall implementeras med hjälp av en eller flera SFCer. Ni skall börja rotera när insignalen ***flytta*** går hög, och när ni är färdiga, skall utsignalen ***färdig*** gå hög. Signalerna ***flytta*** och ***färdig*** är endast höga en scan-cykel. Bordet befinner sig alltid i en arbetsposition när ni får flyttasignalen.

Tips 1: Då signalen ***iLäge*** är true när ni skall börja rotera bordet, så behöver man en delay på minst 0,5 sekunder från start av rotation tills man vet att bordet har åkt ifrån sitt läge och att ***iLäge*** är false.

Tips 2: Då ***startPos*** och ***slutPos*** är insignaler kan dessa inte ändras på. Skapa egna interna variabler om ni behöver det. Definiera egna variabler innan SFCn så man vet vad de är för något.

(6 poäng)