Industriautomation

Tentamen SSY 065, onsdag 22/8, 08:30-12:30, VV

Examinator: Martin Fabian, (772) 3716

Tider för lärarens närvaro: 09:30, 11:30

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 25 poäng. För betygen tre, fyra och fem krävs 10, 15 resp 20 poäng.

Lösningar anslås första vardagen efter tentatillfället på kursens hemsida i Studieportalen.

Tentamensresultaten anslås senast 31/8 på institutionens anslagstavla, kl. 12:30.

Granskning av rättningen får ske 31/8 kl. 12:30 – 13:30 på institutionen.

OBS. Inga hjälpmedel är tillåtna.

Uppgift 1. Robotprogrammering och simulering

a) Förklara vad ett *workobjekt* är och vad det används till. (2p)

Ett workobject är ett koordinatsystem som exempelvis kan läggas i hörnet på ett bord eller inne i en fixtur. Det finns alltid ett world workobject, wobj0. Anledningen till att jobba med andra workobjects än wobj0 är att man kan vilja gruppera flera targets. De placeras då i ett gemensamt workobject. Om man sen flyttar objektet flyttas alla targets med, vilket gör att roboten fortfarande kan hitta alla dessa targets. Exempelvis kan det vara lämpligt att ha ett workobject för häftapparaten, eftersom den då enkelt skulle kunna flyttas, utan att man behöver ändra koordinaterna för de tre häftpunkterna.

b) Förklara vad ett *target* är. (1p)

Ett target är en koordinat med en position och en orientering (samt en konfigurering), exempelvis en hemmaposition för en robot.

c) Förklara vad en *path* är. (1p)

En path är en sekvens av rörelseinstruktioner (t.ex. MoveL, MoveJ) längs vilken roboten rör sig till sina targets.

d) Beskriv för- och nackdelar med offlineprogrammering gentemot onlineprogrammering. (2p)

Fördelar:

Minimerat robotstillestånd

Kraftfulla simuleringsmöjligheter

Ett programmeringssystem för olika robottillverkare

Undviker kollisioner vilket medför bättre arbetsmiljö för operatörerna.

Nackdelar:

Kräver kalibrering

Dyra

Inlärningssvårigheter

Uppgift 2. Flödessimulering

a) Vad är syftet med flödessimulering? (2p)

Testa:

_ Olika layouter

_ Olika processtider

_ Olika egenskaper

_ Eliminera flaskhalsar

_ Kassationer

Dvs variera parametrar som skall undersökas enl problemformulering / Detaljeringsgrad

Nya problemområden kan uppdagas

- Skapar ordning och reda
- Pekar åt rätt riktning
- Tidssparande

- Beslutsstöd (utveckling /styrningen)
- Samlingsplats för idéer/kommunikation
- Övning/träning.

b) Vad menas med en *warmup*? (1p)

- Används för att ta bort data som inte är relevant för helheten
- Används inte i system med återkallande status; ex bank, service centraler
- Warm-up är den tid det tar för modellen att ”svänga” in sig (Steady state)

c) Vilka är de huvudsakliga medlen man har för att påverka resultatet av en flödessimulering? Förklara på vilket sätt dessa påverkar resultatet. (2p)

Användning av buffrar vilket leder till att osäkerheter i en maskin inte påverkar en efterföljande lika mycket.

Arbeta med batcher vilket gör att man minskar omställningstider.

Titta på vilken rutt produkterna tar vilket gör att maskiner som lämpar sig bäst för en viss produkt används

Svar finns i robottext sid 433. (och nedan)

Det största problemet med grafiska CAR-system är att roboten inte intar de OLP-genererade positionerna med erforderlig noggrannhet. Det finns två olika sätt att definiera robotens noggrannhet. Dessa är absolutnoggrannhet, robotens förmåga att inta ett specifikt *xyz*-värde, och repeternoggrannhet, robotens förmåga att repetitivt inta en given punkt (se figur 12.3). Eftersom dagens industrirobotar vanligtvis har mycket

bättre repeternoggrannhet än absolutnoggrannhet är det den sistnämnda som ligger till grund för problemet med noggrannheten i de OLP-genererade positionerna. Typiska repeternoggrannheter är omkring 0.1 mm medan absolutnoggrannheter är i storleksordningen 1 – 3 mm.

Figur 12.3 Beskrivning av repeter- och absolutnoggrannhet.

Detta medför inget problem vid användandet av teach-in metoden, då man med hjälp av det mänskliga ögat kan positionera roboten i rätt läge. Här är det alltså repeternoggrannheten som är mycket viktig för att roboten gång efter gång ska inta den önskade positionen.

Däremot när något off-line system används för att numeriskt generera robotens positioner blir absolutnoggrannheten väldigt viktig. Dessa system kräver att roboten intar en given punkt med tillfredställd noggrannhet för att kunna användas i den utsträckning som det är tänkt. Går inte detta måste ändå roboten tas i anspråk för att omprogrammera alla positioner och off-line programmeringen förlorar sitt syfte.

Uppgift 3. Kommunikation

Förklara OSI modellen för datorkommunikation. (3p)

See the lecture notes.

Uppgift 4. PLC

a) Beskriv de huvudsakliga delarna av en modern PLC. (2p)

I/O-enhet (med in- och utgångsmoduler, plus buffertminne), processor (CPU), programminne (RAM), operativsystemminne (ROM).

b) Förklara den principiella idén bakom *scan-cykel* konceptet som (så gott som) alla moderna PLCer använder? (2p)

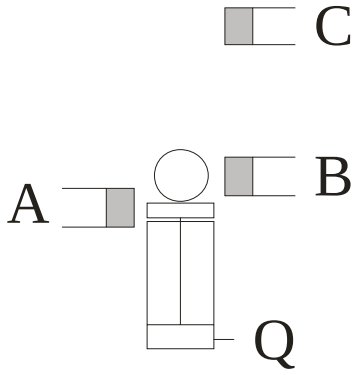
Alla insignalers värde kopieras till buffertminnet, hela programmet exekveras med insignalernas sparade värde och utsignalernas nya värde sparas i utsignalsbufferten. Därefter kopieras utsignalbuffertens innehåll till de egentliga utsignalerna. Denna läs-exekvera-skriv-cykel upprepas om och om igen så länge PLC:n är påslagen, och är det som kallas för *scan-cykel*. Detta görs för att simulera det parallella beteende som reläkopplingar och fast trådad logik uppvisar. Från en utomstående betraktare verkar det som om alla utsignaler ändrar sina värden beroende på insignalerna (och interna minnen), samtidigt.

c) Get exempel på problem som kan inträffa om scan-cykeln inte är kort i förhållande till den styrda processens tidskonstanter. (2p)

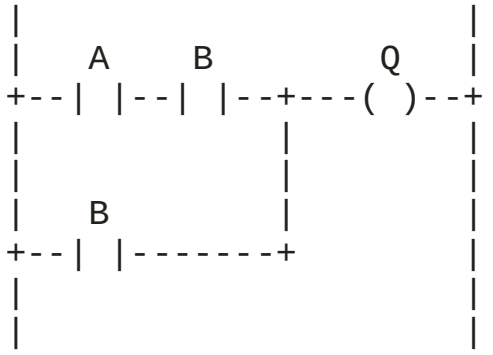
Under exekvera-skriv-delen av scan-cykeln registrerar PLC:n inga insignalsändringar. Det kan medföra att de genererade utsignalerna är felaktiga i förhållande till det aktuella tillståndet (vilket alltså då är skiljt från det tillstånd PLC:n registrerat) hos den styrda processen.

Uppgift 5. LD programmering

Emil och Emilia skulle programmera en PLC att styra ett enkelt labsystem, se bilden nedan till vänster. De skrev LD-koden till höger nedan och förväntade sig att när kolven var nere och kulan var på plats så skulle kolven gå upp till sitt övre läge, vid C.



Det enkla labsystemet. A, B och C är givarsignaler, A talar om att kolven är nere, B talar om att kulan är på plats, C talar om att kolven är uppe. Q är utsignal som, då den är hög, kör upp kolven.



Ladderdiagrammet som försöker styra kolven till sitt övre läge vid C när kolven är nere och kula är på plats.

Tyvärr fick de ett helt annat beteende än det tänkta. Beskriv detta beteende, tala om vad som var problemet och föreslå en lösning. (5p)

The program represents the logic expression `Q := AB + B`, which is equivalent to `Q := B`. What happens is that as soon as a ball arrives, Q is set high and the cylinder moves up. However, after a while B no longer signals, which sets Q low and thus the cylinder moves down again. Once it moves into the region of sensor B, Q is set high again and the cylinder moves up. What we get is an oscillating behaviour due to the scan-cycle property of the PLC. A solution must include a memory to remember that we should continue up even though the B-sensor no longer signals. Then we also need a way to break the memory, X in the LD below.