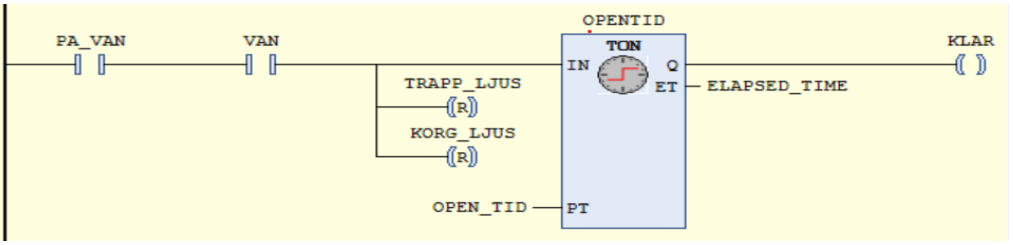
Dokumentation av: Projektrapport	*Datum för senaste version:* 2022-11-30	*Senaste version nr:* Version 1
Ansvarig för dokument: Mehdi Haidari Theodor Ahlgren Grupp 3:G4	*Granskad av:*	Chalmers Tekniska Högskolan

Projektrapport styrsystem för hiss

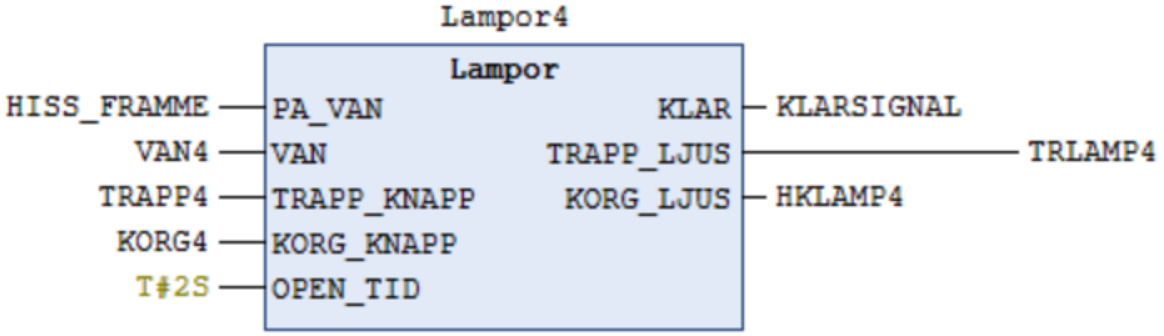
Funktionsblock och POU som innehåller blocket

Vårt funktionsblock, som heter "Lampor", tar emot en beställning som gjorts till våning X via ett knapptryck inifrån korgen eller i trapphuset på våning X används en SET-vippa för att tända lamporna i trapphus och korg, sant hålla dessa lampor tända även då knappen släppts. Lamporna släcks först hissen anlänt till våning X, eftersom att NO-signalerna "PA_VAN" och "VAN" blir true. Signalerna gör en reset på lamporna tillhörande vån X och startar en timer som motsvarar stopp-tiden "OPEN_TID" för korgen på våning X. När den inställda tiden passerat skickas en true signal ut via "KLAR" som indikerar att hissen är redo att fokusera på nästa beställning.



POU:t hissFBD använder sig av funktionsblocket Lampor en gång per våning. I specifikationen till inlämning 3 beskrivs hur SFC:t ska kommunicera med POU:t via flaggor, vilket vi har implementerat i vårt styrsystem. När korgen anländer till en våning sätts den globala flaggan "HISS_FRAMME" till sann från SFC:t. Flaggan används i sin tur i kombination med den globala variabeln för den aktiva våningen i POU:t med lamp-blocken som en signal för att starta timern på den aktuella våningen.

På samma sätt finns även en global flagga som styrs av POU:t och som avläses i form av en transition av SFC:t, "KLARSIGNAL". När SFC:t har nått en våning och kommunicerar till hissFBD via flaggan "HISS_FRAMME" som startar timern, inväntas klarsignalen som tar sekvensen till nästa steg.



Om blocket skulle ha varit i SFC:t, skulle det behövas flera variabel och mer arbete, men via att ha blocket i ett eget program kan man använda signalerna i SFC:t och samtidigt hålla blockets funktionalitet oberoende mellan andra program.

Anledningen till att vi inte fick använda globala variabler i funktionsblocket beror att vi vill ha ett återanvändbart och abstrakt block. Vi vill att blocket ska funka som vilket inbyggt block i programmet som helst, exempelvis en SR-vippa, man ska inte behöva studera hela programmet för att förstå vad blocket lampor har för funktion. Användandet av globala variabler skulle medföra stora ändringar varje gång som en ny våning lades till eller togs bort samt kräva att man höll koll på de globala variablerna och vart dom kommer ifrån.

Insignaler	**Typ**	**Beskrivning**
PA_VAN	BOOL	
VAN	BOOL	VAN skickar in en flagga som är true om hissen är på våning VAN.
TRAPP_KNAPP	BOOL	Knapp från trapphuset.
KORG_KNAPP	BOOL	Knapp inifrån hisskorgen.
OPEN_TID	TIME	Tiden som hissen väntar på våningen.

Utsignaler	**Type**	**Beskrivning**
KORG_LJUS	BOOL	Lamp in i hisskorgen.
TRAPP_LJUS	BOOL	Lamp i trapphuset.
KLAR	BOOL	Klarsignal att hissen kan åka till nästa beställning.

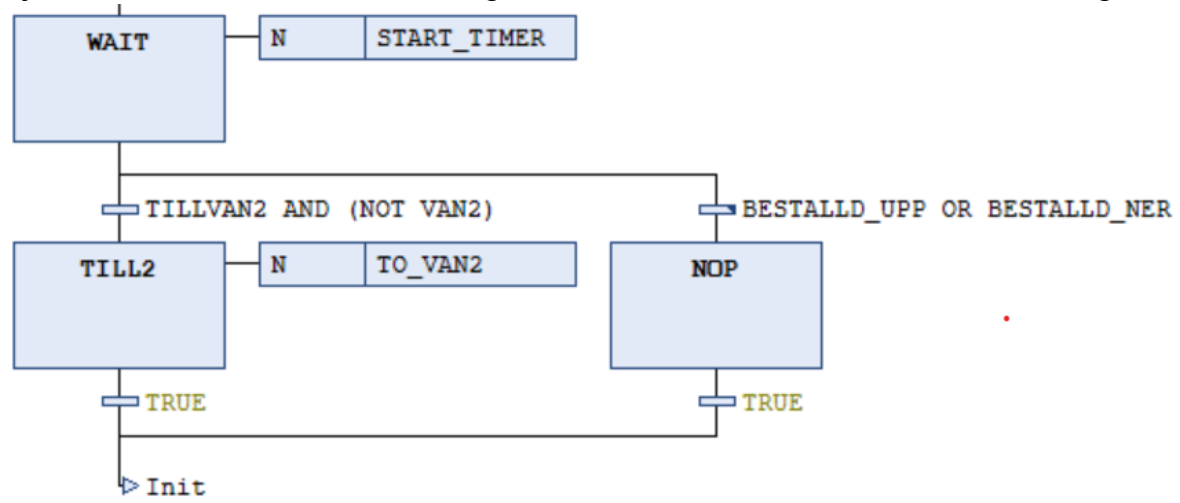
Hissekvens i SFC

SFC:t är huvuddelen av vårt program och det är här som hissens motorer styrs. Här finns huvud-stegen "uppat" och "nerat" som medans de är aktiva även sätter de globala variablerna för hissmotorerna upp respektive ner, till true via actions. För att nå dessa steg krävs att någon av att hissen beställt antingen upp eller ned, som finns i vårt SFC som transitions.
Sekvensen befinner sig i steget "uppat" eller "nerat" tills att korgen är på en våning som har lamporna tända (vilket är en transition). Därefter skickas signalen till POU:t om att korgen är framme på en våning och nästa transition, som beror på klarsignalen från POU:t, inväntas.

En action är ett "kommando" i SFC:t och en transition är ett villkor för en övergången. En action kan exempelvis vara att sätta en flagga, medan en transition är villkoret för en övergång. När ett villkor inträffar övergår sekvensen till nästa steg.

Prioriteten mellan korgens upp och ner beställningar löser vi i ett separat POU och med två globala variabler: "PRIO_UPP" och "PRIO_NED". Endast en utav prioriteringsvariablerna kommer vara aktiv åt gången, och när den ena stängs av sätts den andra på. Från start är "PRIO_UPP" true. Då korgen når en våning som inte har några tända våningar ovan, växlar prioriteten till att prioritera nedanstående våningar, vilket är fallet tills dess att korgen befinner sig på en våning som inte har några tända våningar under sig, och uppåt-prioriteringen aktiveras återigen.

För att få styrsystemet att återvända till våning 2 efter en viss tid utan beställningar byggde vi på SFC:t. Där sekvensen tidigare börjat om SFC-loopen lade vi istället till ett steg som startar en timer. Efter detta steg finns två parallella steg med varsina transitions. Den vänstra transitionen blir aktiveras då tiden från den nyligen aktiverade timern når 1,5s och leder till ett steg som sätter igång lamporna på våning 2. Följande transition är alltid true, vilket betyder att när en beställning på våning 2 lagts hoppar programmet tillbaks till sitt normala "ta-emot-beställningar"-läge, med en beställning på våning 2.
Den högra transitionen aktiveras om en beställning görs, och det följande steget samt den följande transitionen har ingen funktion, utan körs rätt igenom.

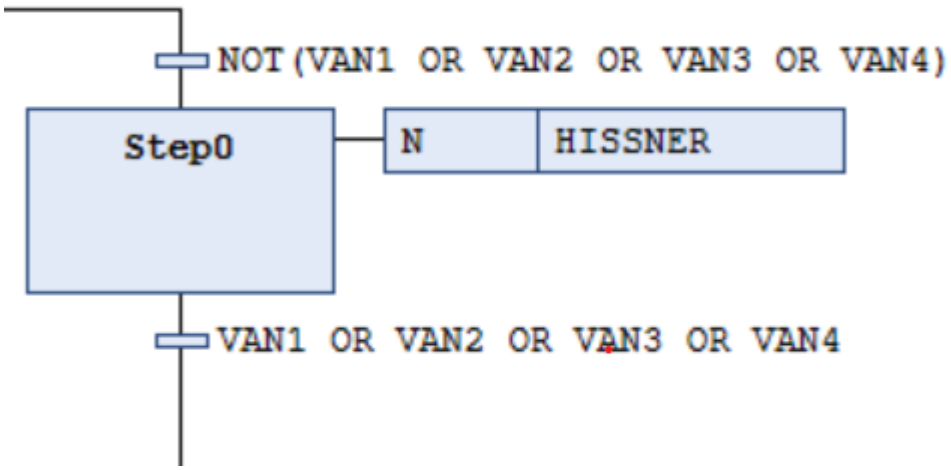


TILLVAN2 är den globala variabel som sätts true av timern aktiverad av action "START_TIMER".
TO_VAN2 är den globala variabel som aktiverar lamporna på våning 2.

Hur löste ni extrauppgiften att hissen kunde starta mellan två våningar?

För att möjliggöra start mellan två våningar placerade vi ut ett tredje steg parallellt med stegen "Uppat" och "Nedat". Vi satte en transition innan steget som var aktivt då korgen inte

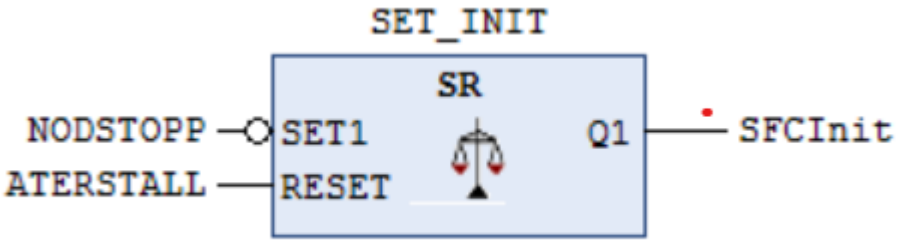
befann sig på någon våning samt en transition efter steget som var aktiverades då korgen hade körts till en våning. Själva steget har en action som startar hissmotorn som kör hissen nedåt. Alltså om korgen befinner sig mellan våningar så körs den nedåt tills det att en våning har nåtts. Eftersom det nya steget ligger längst åt höger, vilket innebär att transitionen kontrolleras sists, kommer det inte att strula till det när en korg kör från en våning till en beställd våning, trots att korgen befinner sig mellan två våningar.



Nödstopp

Vår hiss har även knappar för nödstopp och återställning placerade i hisskorgen. Nödstoppsknappen är av brytande slag (NC) och när den trycks in stannar den intryckt tills att den fysiskt ställs tillbaka inne i hisskorgen. Eftersom att nödstopp är av typen NC kommer ett knapptryck på nödstopp bryta den aktuella sekvensen.

Vår globala variabel för nödstopp kopplas med en negation till set-delen av en SR-vippa. Detta eftersom nödstopp alltid är sann till dess att knappen trycks in. Till reset delen kopplas den globala variabeln för återställsknappen och till Q1 kopplas SFCInit. Eftersom att vippan är S-dominant krävs att nödstoppsknappen först dras ut, sedan kan återställ användas för att göra en reset som återställer programmet.



En nödstoppsknapp bör alltid designas som NC på grund av säkerhetsskäl. Nödstopp ska kunna användas för att stoppa anläggningen även om PLC-programmet skulle vara felaktigt, därför kopplas nödstoppet så att spänningen bryts till komponenterna som ska stoppas.

I labbet kopplades nödstoppet till källan, hissen och plc:n. Spänningen (24 V) leddes via nödstoppet till både hissen och plc:ns in-och utgångar. När nödstopp var intryckt ledes ingen spänning. Detta gör att nödstoppet ligger utanför programmet. En annan signalkabel kopplades också från nödstoppet till plc:n för att ta information om nödstoppets status. .

Att man har en återställningsknapp tillsammans med en nödstoppsknapp beror också på säkerhetsskäl. När ett nödfall inträffat, exempelvis att någon har kommit i kläm, vill vi manuellt kontrollera hissen och undvika att hissen tar nästa beställning och åker till en ny våning. Om vi vill köra hissen manuellt måste vi dra ut nödstoppsknappen och återställningsknappen försäkrar oss då om att programmet inte kommer att köras, tills dess att återställningsknappen har tryckts.

Fysiska kopplingar

För att identifiera in- och utsignaler till PLC använde vi oss av modulen på PLC:t. Via knapptryck från den fysiska hissen noterade vi vilka adresser på modulen som lös upp. Senare i labben kopplades adresserna till variabler i vårt program. De slutgilltiga kopplingar som gjordes i labbet för att den fysiska hissen och PLC skulle fungera var följande:

Fysiska insignaler	**Type**	**Beskrivning**	**Adress**
TR_K1	BOOL	Knapp för våning 1 i trapphuset.	%IX0.0
TR_K2	BOOL	Knapp för våning 2 i trapphuset.	%IX0.1
TR_K3	BOOL	Knapp för våning 3 i trapphuset.	%IX0.2
TR_K4	BOOL	Knapp för våning 4 i trapphuset.	%IX0.3
HK_K1	BOOL	Knapp in i hisskorgen för våning 1	%IX1.0
HK_K2	BOOL	Knapp in i hisskorgen för våning 2	%IX1.1
HK_K3	BOOL	Knapp in i hisskorgen för våning 3	%IX1.2
HK_K4	BOOL	Knapp in i hisskorgen för våning 4	%IX1.3
GLG_VAN1	BOOL	Gränslägesgivare för våning 1	%IX0.4
GLG_VAN2	BOOL	Gränslägesgivare för våning 2	%IX0.5
GLG_VAN3	BOOL	Gränslägesgivare för våning 3	%IX0.6
GLG_VAN4	BOOL	Gränslägesgivare för våning 4	%IX0.7
NOD_KNAPP	BOOL	Nödstopp knappen för nödläge, har hög signal när den inte är	%IX1.4

		intryckt.	
ATER	BOOL	Knapp för återställning av nödläge	%IX1.5

Fysiska utsignaler	**Type**	**Beskrivning**	**Adress**
TR_L1	BOOL	Lampa för våning 1 i trapphuset	%QX0.4
TR_L2	BOOL	Lampa för våning 2 i trapphuset	%QX0.5
TR_L3	BOOL	Lampa för våning 3 i trapphuset	%QX0.6
TR_L4	BOOL	Lampa för våning 4 i trapphuset	%QX0.7
HK_L1	BOOL	Lampa för våning 1 in i hisskorgen	%QX0.0
HK_L2	BOOL	Lampa för våning 2 in i hisskorgen	%QX0.1
HK_L3	BOOL	Lampa för våning 3 in i hisskorgen	%QX0.2
HK_L4	BOOL	Lampa för våning 4 in i hisskorgen	%QX0.3
MOTOR_UPP	BOOL	Motor som kör hissen uppåt	%QX1.0
MOTOR_NER	BOOL	Motor som kör hissen nedåt	%QX1.1

Nödstopps signalen hade en hög signal (24 V) som kopplades till plc:s in-och utgångar plus en annan signal från nödstoppet till en ingång på plc för att informera om nödstoppets status. Genom att ha nödstoppet som spännings förare till hissen och plc:n garanterar man att nödstoppet är oberoende av programmet. Nödstoppet stoppar processen genom att inte mata plc:n och hissen med spänning. Då vet man att nödstoppet verkligen stoppar systemet.