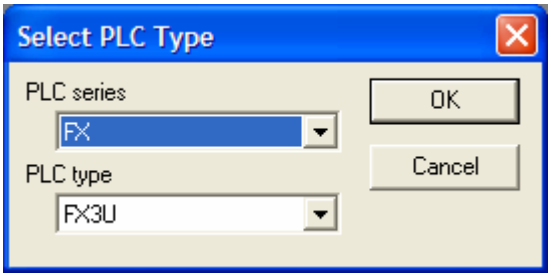
070822/ Thomas Munther
IDE-sektionen, Halmstad Högskola

Laboration 1 Styrteknik (preliminär version)

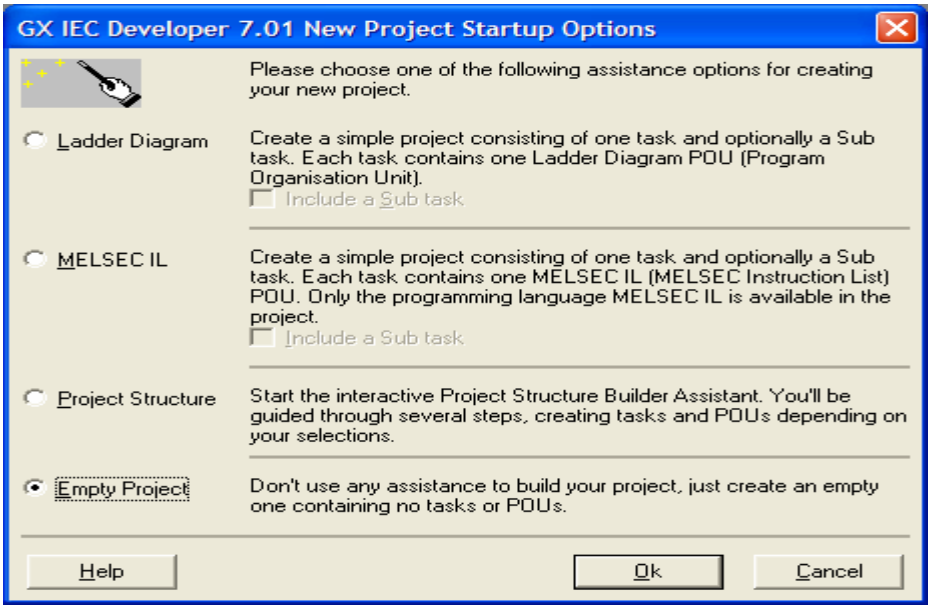
Starta upp programmet **GX IEC Developer** ligger på skrivbordet eller alternativt under program.

Gå sedan in under **Project- > New**.
Välj vilken PLC-serie respektive vilken modell du skall arbeta mot. I vårt fall:
FX-serie och modelltyp **FX3U.** Se nedan hur det ser ut.



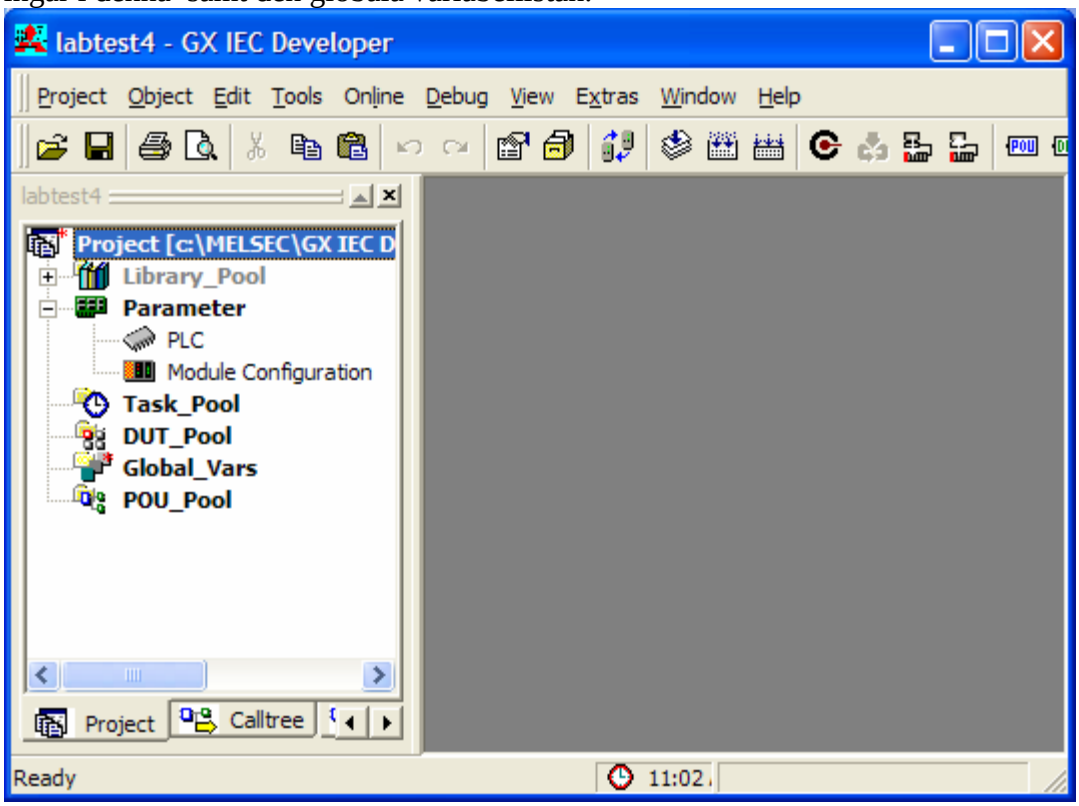
Du får nu ett förslag var du skall lägga ditt än så länga tomma projekt:
Skapa en underkatalog **exempel** till defaultvalet t e x
C:\MELSEC\GX IEC Developer 7.01.

Direkt efter så får du uppmaning att välja hur projektet skall skrivas.
En ruta som heter **New Project Starting Options** dyker upp med fyra alternativ.
Välj **Empty Project** ! Se figur 1!



Figur 1

Nu öppnar sig en list längst till vänster en trädstruktur av ditt lilla projekt.
Detta brukar kallas för projektnavigatorn. Den finns öppen och åtkomlig under hela tiden. De viktigaste grenar som du behöver använda varje gång är **Task_Pool, Global_Vars** och **POU_Pool**. I Task_Pool lägger du olika programmeringsuppgifter som du har löst, men dessa sätts samman av olika programdelar POU:s. För varje Task måste anges vilka POU:s som ingår. I Global_Vars deklareras dina globala variabler (ingångar och utgångar på PLC och andra variabler som används i flera POU. När du skall skicka över ett program till PLC:n så är det en Task som först måste kompileras och de POU som ingår i denna samt den globala variabellistan.



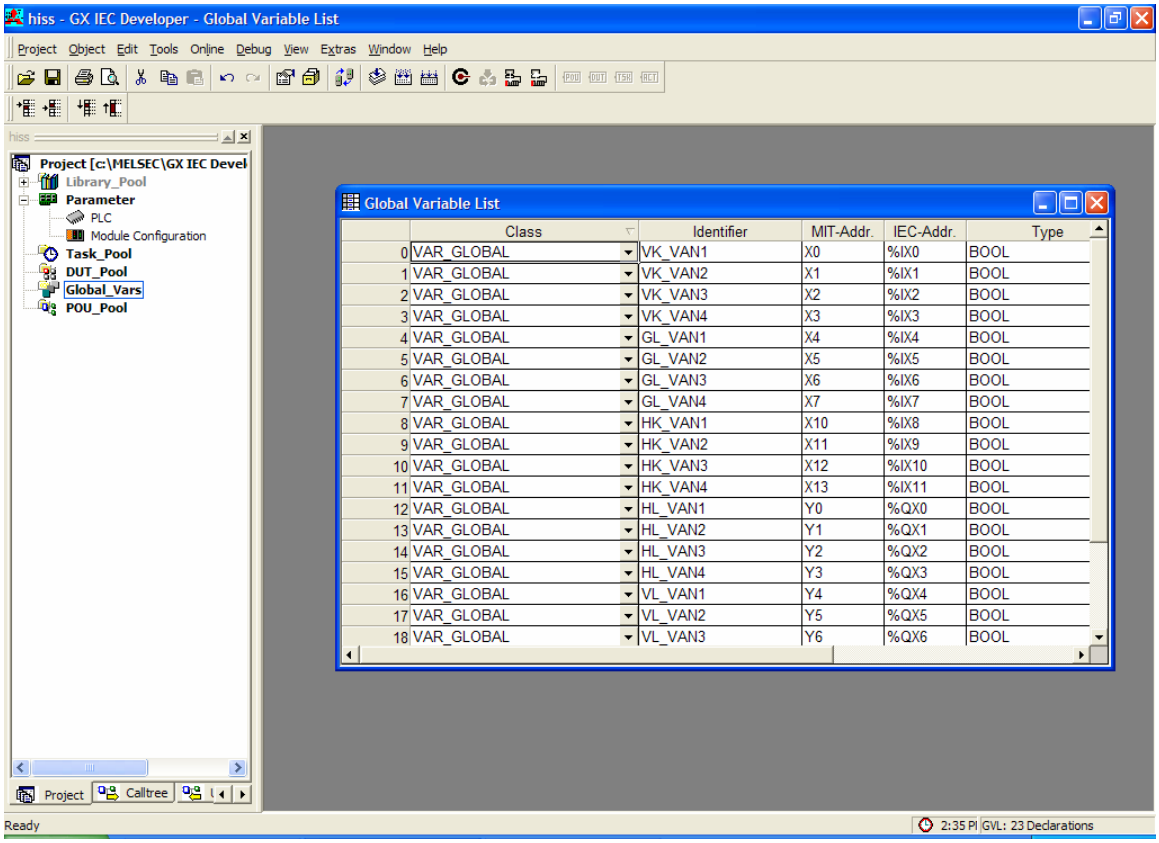
Figur 2

Meningen med laborationen är att vi skall försöka lära oss att använda programmet för att göra en enkel tillämpning. Vi skall försöka göra en enkel hisstyrning av en trevåningshiss och använda lite olika sätt att programmera en PLC t ex ladderprogram(reläprogram) och SFC – Sequential Function Chart (Grafcetliknande program).

Markera **Global_Vars** och öppna denna ! Här skall vi lägga in våra globala variabler.
Det betyder att i vårt fall lägger vi in insignaler (ingångar)till PLC respektive utsignaler från denna (utgångar). Typiska insignaler kan vara gränslägesgivare och tryckknappar.
Gränslägesgivare ger en 1:a om hissen ligger framför dess våning.
Tryckknappen ger en 1:a om den är intryckt , en 0:a annars.
Utsignaler blir saker och ting som vi kan påverka på vår hissmodell som lampor och hissmotorn.
Lägg nu in följande:

GL_VAN1 (gränslägesgivare våning 1) under **Identifier,** adressen (**MIT-adress**) blir på Mitsubishi systemet FX3U X0 och **Type** blir BOOLEAN. Resten behövs inte fyllas i. Lägg nu till gränslägesgivare GL_VAN2 och GL_VAN3 på samma sätt i variabellistan.

För att lägga till fler variabler i listan så trycker vi på symbolen (Insert After) under disketten. Automatisk tilldelning.
Min tanke är att vi lägger in alla in- och utgångar som kommer till och från hissmodellen på en gång som vi kommer att använda. Gör som ovan !



Figur 3

Vi bestämmer själva namnen på identifierarna. Undvik svenska bokstäver å,ä och ö.

Global variabellista och oktal numrering.
Först följer ingångar och dessa inleds med ett X.

Class	**Identifier**	**MIT-Addr**	**IEC-Addr**	**Type**
VAR_GLOBAL	VK_VAN1	X0	%IX0	BOOL
VAR_GLOBAL	VK_VAN2	X1	%IX1	BOOL
VAR_GLOBAL	VK_VAN3	X2	%IX2	BOOL
VAR_GLOBAL	VK_VAN4	X3	%IX3	BOOL
VAR_GLOBAL	GL_VAN1	X4	%IX4	BOOL
VAR_GLOBAL	GL_VAN2	X5	%IX5	BOOL
VAR_GLOBAL	GL_VAN3	X6	%IX6	BOOL

VAR_GLOBAL	GL_VAN4	X7	%IX7	BOOL
VAR_GLOBAL	HK_VAN1	X10	%IX10	BOOL
VAR_GLOBAL	HK_VAN2	X11	%IX11	BOOL
VAR_GLOBAL	HK_VAN3	X12	%IX12	BOOL
VAR_GLOBAL	HK_VAN4	X13	%IX13	BOOL

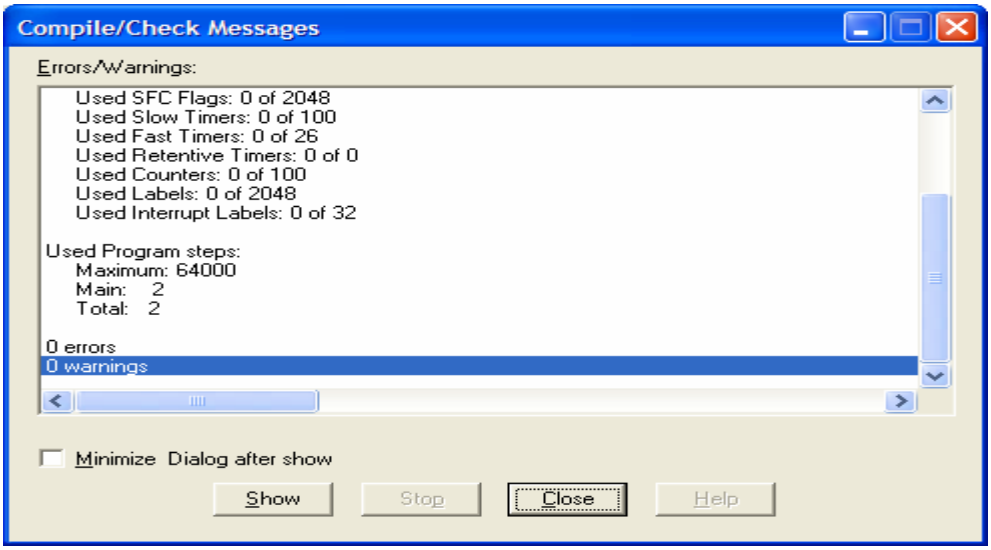
Sedan följer utgångar med oktala adresser som föregås av ett Y.

Class	**Identifier**	**MIT-Addr**	**IEC-Addr**	**Type**
VAR_GLOBAL	HL_VAN1	Y0	%QX0	BOOL
VAR_GLOBAL	HL_VAN2	Y1	%QX1	BOOL
VAR_GLOBAL	HL_VAN3	Y2	%QX2	BOOL
VAR_GLOBAL	HL_VAN4	Y3	%QX3	BOOL
VAR_GLOBAL	VL_VAN1	Y4	%QX4	BOOL
VAR_GLOBAL	VL_VAN2	Y5	%QX5	BOOL
VAR_GLOBAL	VL_VAN3	Y6	%QX6	BOOL
VAR_GLOBAL	VL_VAN4	Y7	%QX7	BOOL

Hissen har två utgångar för styrning i vardera riktningen. Dessa skall inte aktiveras samtidigt. Därefter skall vi använda specialadress som ger en 1 Hz puls ut växlande mellan värde 0 och 1.

Class	**Identifier**	**MIT-Addr**	**IEC-Addr**	**Type**
VAR_GLOBAL	Motor_upp	Y10	%QX10	BOOL
VAR_GLOBAL	Motor_ner	Y11	%QX11	BOOL
VAR_GLOBAL	PULS_1Hz	M8013	%MX0.8013	BOOL

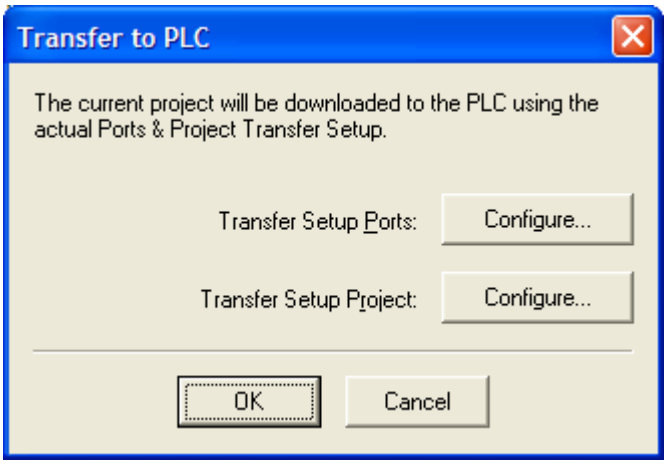
Notera att förkortningar VK_VAN1 står för våningsknapp 1 och VL_VAN1 betyder våningslampa 1. Dessa hittar man vertikalt hos hissmodellen.
Förkortningar HL_VAN1 respektive HK_VAN2 står för Hisslampa våning 1 respektive Hissknapp våning 2. Dessa hittar du liggande på bottenvåningen i hissen. De är egentligen tänkta att sitta inne i hissen, men av praktiska skäl har dessa lagts utanför. Se figur 26 !
Så här långt har vi endast skapat en lista med namn kopplade till adresser hos vårt PLC-system. Faktum är att vi kan testa och föra över denna lista till PLC:n.
Gå in under **Project-> Rebuild all !** Om kompileringen nu gick bra så fås:



Figur 4

Här kan man se att vi fick inga fel och inga varningar. Ni kan också vilken prestanda som PLC, d v s antal counters, timers, antal programsteg, antal tillståndssteg o s v.
Skulle vi ha fått fel så måste dessa åtgärdas innan vi kan skicka över programmet till PLC:n. Varningar däremot går att bortse ifrån och ändå överföra ett program.
I vårt fall är det ju inget riktigt program utan endast variabeldeklarationer.

Nästa steg blir att föra över vår globala variabellista till PLC:n och detta görs enligt: **Project-> Transfer-> Download to PLC.** Givetvis måste seriekabeln vara inkopplad mellan dator och PLC-system. Sätt även PLC i RUN-läge. Det finns en switch jämte seriekablaget på PLC.



Klicka OK!

Därefter får du en fråga om du vill sätta PLC i STOP-läge. Gör det !
Överföringen startat om allt gick bra så får du på nytt en fråga att sätta PLC:n i RUN-läge.

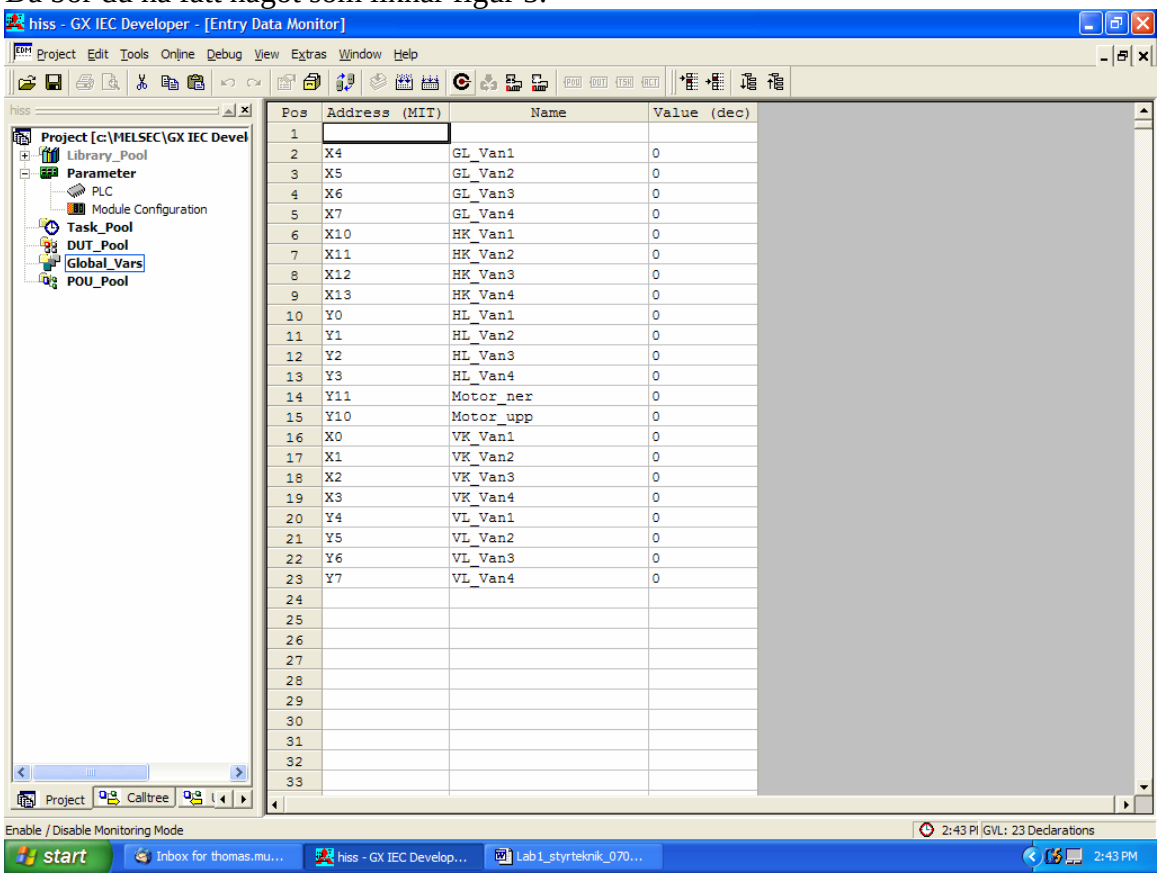
Monitorering/Övervakning av ingångar och utgångar

Hissmodulen skall vara inkopplad till PLC-plattan via flatkablar. Glöm inte att mata hissen med 24 Volt likspänning.
Nu skall vi se om vi kan visa status på ingångarna till hiss och PLC:n, d v s i vårt fall knappar och gränslägesgivare samt även se om vi kan påverka hissmodulens lampor samt få hissen och röra på sig.

Gå till **Online-> Entry Data Monitor**

Om tabellen är tom . Markera med musen i densamma. Högerklicka och välj **Insert Objects.** Nu dyker det upp en objekt lista där du kan välja vad du vill övervaka.
Välj **Global Variables** och **Add all** !

Då bör du ha fått något som liknar figur 5.



Pos	Address (MIT)	Name	Value (dec)
1			
2	X4	GL_Van1	0
3	X5	GL_Van2	0
4	X6	GL_Van3	0
5	X7	GL_Van4	0
6	X10	HK_Van1	0
7	X11	HK_Van2	0
8	X12	HK_Van3	0
9	X13	HK_Van4	0
10	Y0	HL_Van1	0
11	Y1	HL_Van2	0
12	Y2	HL_Van3	0
13	Y3	HL_Van4	0
14	Y11	Motor_ner	0
15	Y10	Motor_upp	0
16	X0	VK_Van1	0
17	X1	VK_Van2	0
18	X2	VK_Van3	0
19	X3	VK_Van4	0
20	Y4	VL_Van1	0
21	Y5	VL_Van2	0
22	Y6	VL_Van3	0
23	Y7	VL_Van4	0
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			

Figur 5

Starta monitorering genom att klicka på symbolen som liknar en piltavla i menyn. Dubbelklicka på en utgångsvariabels Value så ändrar den sitt värde och om allt är rättkopplat så kan du säkert se/höra någon förändring på hissmodulen.

Upprepa samma sak fast för en ingångsvariabel. Händer det något ?

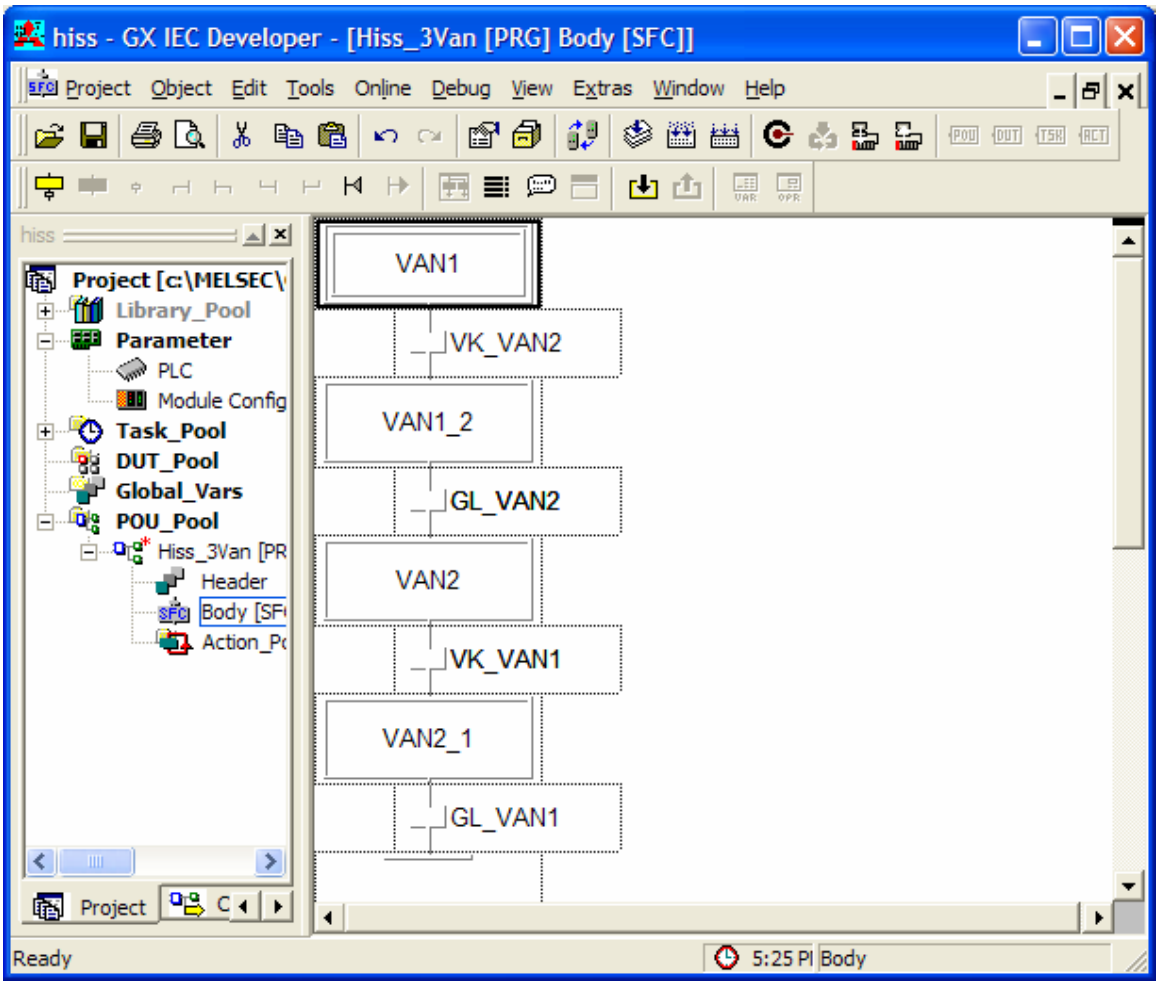
Ingångarna tar ju emot signaler utifrån som en knapptryckning eller att vi står framför en våning med hissen. Så det spelar ju inte så stor roll att vi ändrar värdet inuti **Entry Data Monitor.** Tryck på hissknapparna/våningsknapparna istället eller kör hissen till en våning. Ses på baksidan av hissmodulen om vi ligger på en våning. Indikeras med en tänd lysdiod.

Efter att ha lekt en stund stäng av monitorering (klicka på piltavlan i menyn).
Klicka ner **Entry Data Monitor** tabellen. Nu är det dags att skriva ett litet hissprogram.

PLC-program

UPPGIFT:
Vi skall i laborationen konstruera en hiss som kan åka från våning 1 till våning2 och tillbaka. Indikera med respektive våningslampa om hissen befinner sig där.
Den bör kunna räkna hur ofta den har besökt en våning samt ha tidsfördröjningar på våningsknappar och våningslampor

Jag visar hur en tvåvåningshiss kommer att se ut implementerad i SFC. Skulle ni tappa bort er i handledningen gå tillbaka hit för att komma tillbaka in på spåret.



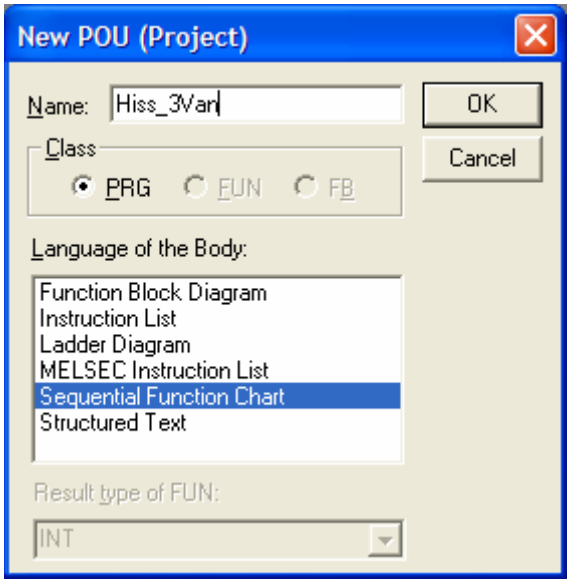
Figur 6

Vårt SFC-program innehåller 4 stycken olika symboler första rutan längst upp är en dubbeltecknad rektangel. Det betyder att det är initialtillståndet måste finnas en sådan ruta. När PLC:n startar hamnar vi alltid där. Längst ner finns en horisontell linje. Denna är **final step(** måste finnas**)**. Den anger när sekvensen är slut och vi kommer då tillbaka till initialtillståndet och sekvensen startar om. Däremellan finns vanliga tillståndsrutor(**Step**).
Vi bestämmer vad dessa skall heta. Ge dem vettiga namn och undvik å,ä och ö.
Mellan varje tillstånd finns alltid ett övergångsvillkor (**Transition**). Villkoret anger vad som måste vara uppfyllt för att lämna ett tillstånd och gå till ett annat.

Hur skall vi beskriva vad som sker i vårt ovanstående funktionsdiagram. Se figur 6 !
Antag att hissen står på våning 1. Hur länge skall den stå där ?
Ja tills någon trycker på våningsknapp 2. Då blir övergångsvillkoret sant och vi lämnar tillstånd Van1 och går till tillstånd Van1_2. Hur länge skall vi befinna oss i detta tillstånd. Tja tills hissen befinner sig på våning2(GL_VAN2 ger en 1:a) och då kommer vi till tillstånd Van2 o s v. Vi har ovan beskrivit hur vi tänker oss hur funktionsdiagrammet beskriver hissförloppet, men vi måste också få något att hända.

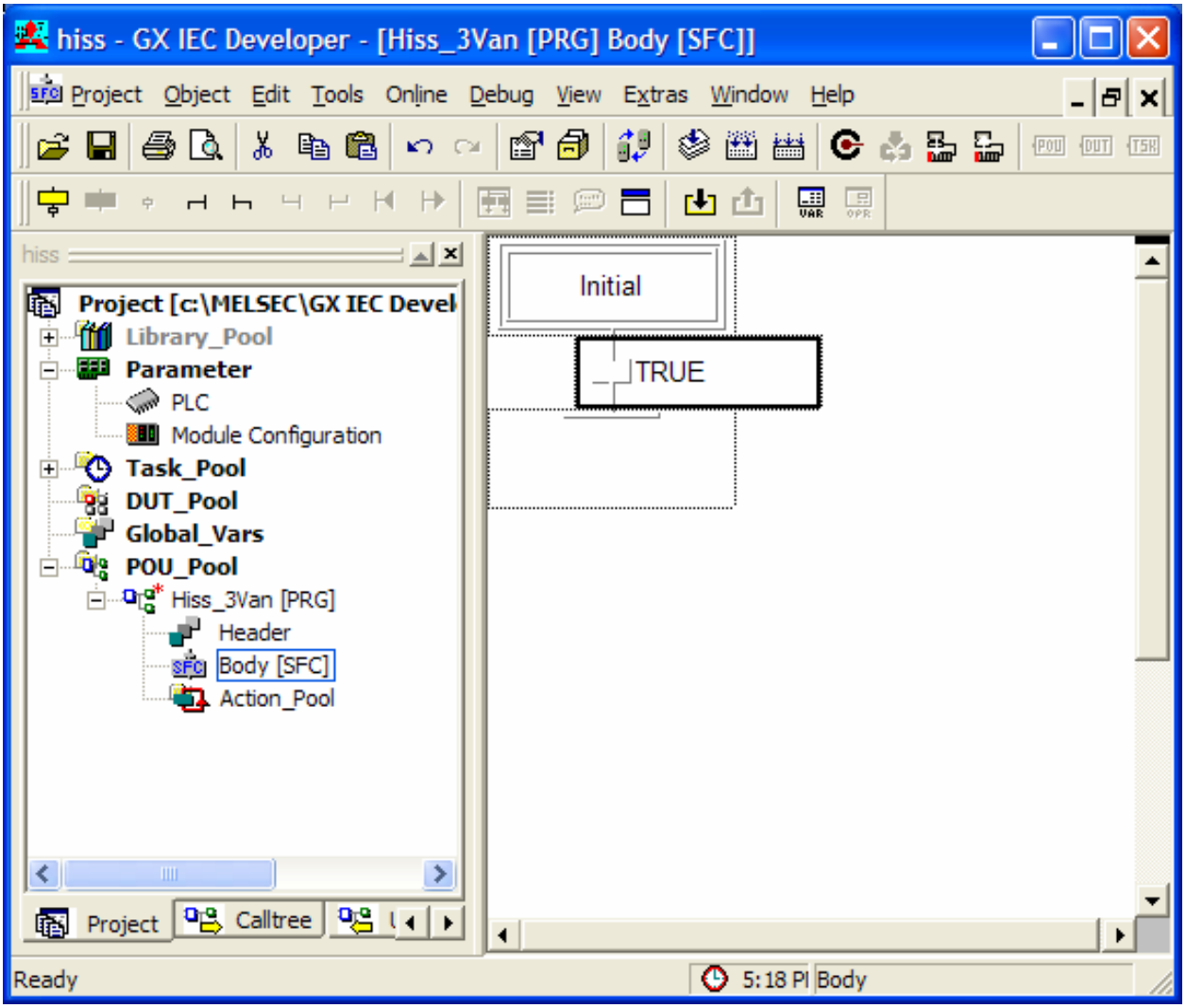
Våningslamporna kanske skall indikera på vilken våning befinner sig. Hissen måste ju faktiskt påverkas att köra upp och ned. Kort sagt vi måste knyta aktiviteter/händelser (**Actions**) till tillstånden annars händer inget. Ni får endast försöka påverka statusen hos utgångar och minnen i själva tillstånden aldrig i ett övergångsvillkor.

Programmeringspråket som vi skall använda kallas för Sequential Function Chart (SFC) det väljer vi genom att gå in i Project Navigator under **POU Pool. Markera denna och välj New Pou. Finns även i menyn.**



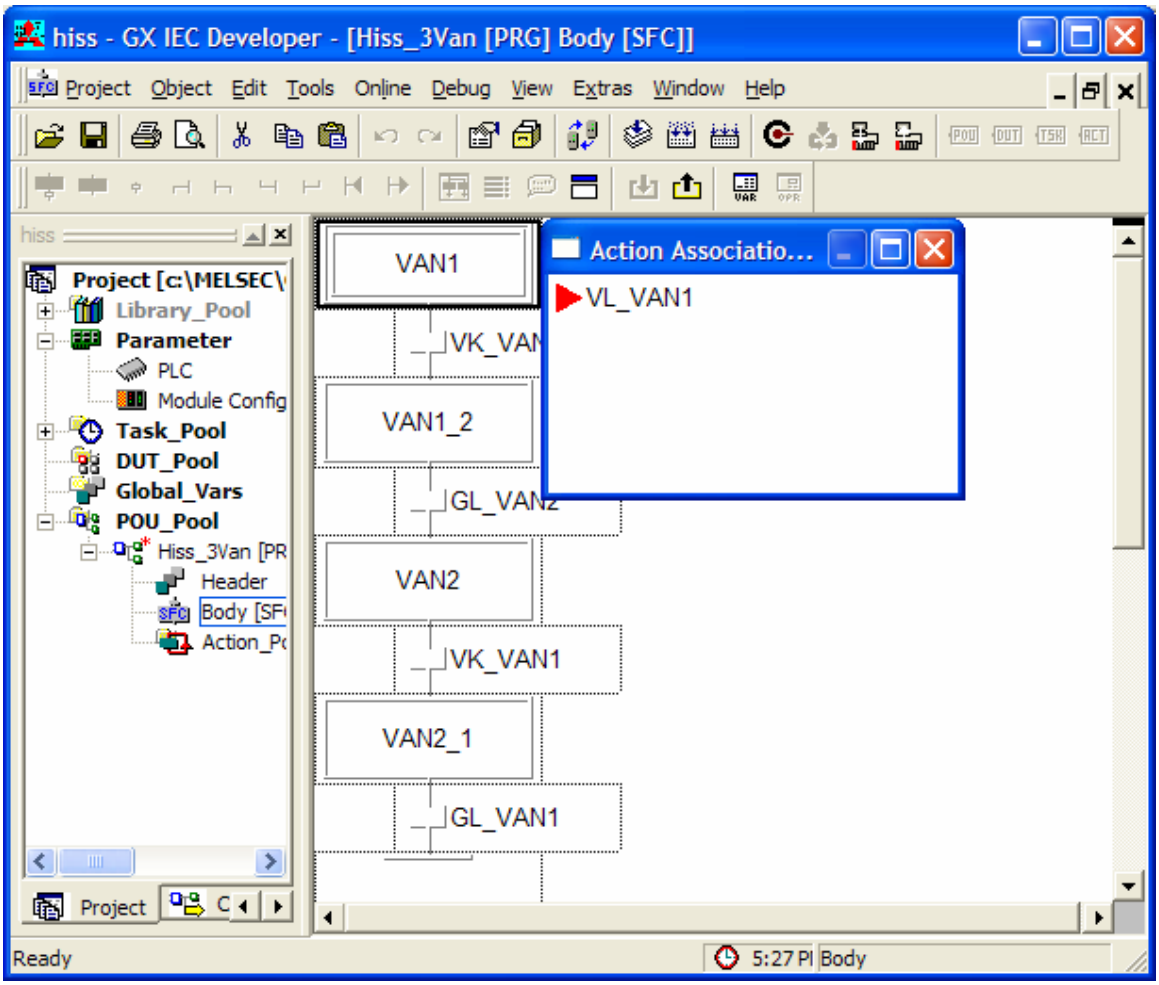
Döp denna till Hiss_3Van. Välj **Class** till PRG och sedan programspråk Sequential Function Chart (SFC) för själva programkroppen (Body).
Du har nu skapat en POU-Program Organizer Unit (programdel) och du har också valt vilket språk som du vill använda.
I navigatorn under POU (om du klickar på denna) finns nu tre grenar **Header, Body** och **Action Pool**. Header är lokal variabelista. Här deklareras de variabler som används i POU:n och som inte är globala. Body är i vårt fall själva SFC-kroppen och slutligen Action Pool innehåller aktiviteter som är lite mer komplexa än bara en direkt påverkan av en utgång.

Klicka på Body:n i **Project Navigator.** Då ser det ut som figur 7 nedan.
Den innehåller endast ett initial step, ett final step samt ett övergångsvillkor.
Markera övergångsvillkoretTRUE. Gå sedan upp i symbolmenyn och klicka 3 ggr på den gula symbolen (Step and Transition). Nu har vi fått lika många tillstånd som i figur 6.
Namnge tillstånden enligt figur 6. Skriv också in motsvarande globala variabelnamn för övergångsvillkoren.



Figur 7

Hur skapar vi en action ? Finns olika sätt. Det enklaste är helt enkelt att skriva namnet på den utgång som vi vill påverka i det tillståndet.
Dubbelklicka på tillståndsruta Van1 och skriv i motsvarande **Action association Van1** VL_VAN1. Se nedan i figur 7 !

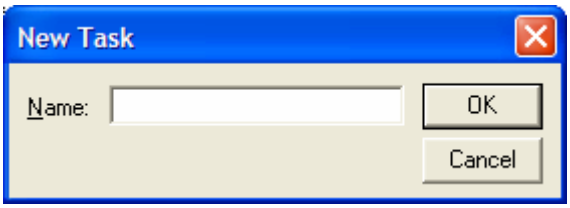


Figur 8

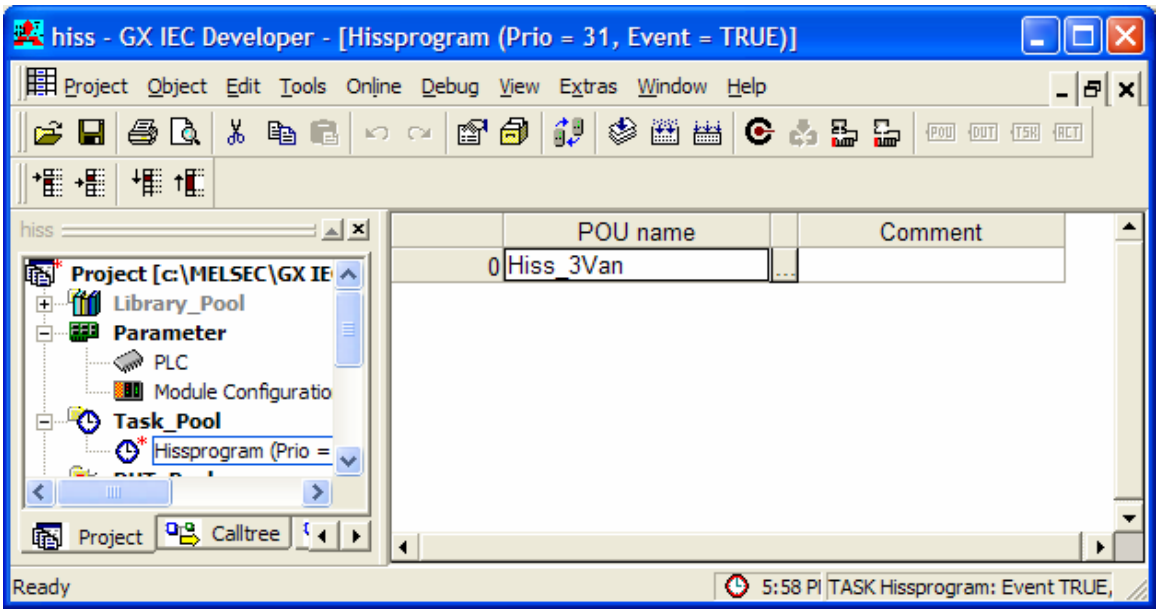
Skriv motsvarande för tillstånden under:

Step	**Action**
Van1_2	Motor_upp
Van2	VL_VAN2
Van2_1	Motor_ner

Faktum är att nu återstår bara en sak innan vi kompilerar programmet och skickar över det till PLC:n för provkörning.
Nämligen de POU vi vill använda i vårt program måste specificeras i en **TASK** och dessa ligger i en **TASK_Pool**. Hur kan vi tolka en en TASK ? Någon form av administrerande funktion som samordnar de POU vi använder (vi har bara en just nu).
Markera TASK_Pool och välj **New Task**. Döp denna till något: Hissprogram



Markera Hissprogram under TASK_Pool och dubbelklicka på denna.
Lägg in vilka POU som ingår i din TASK Hissprogram. Se nedan figur 9!



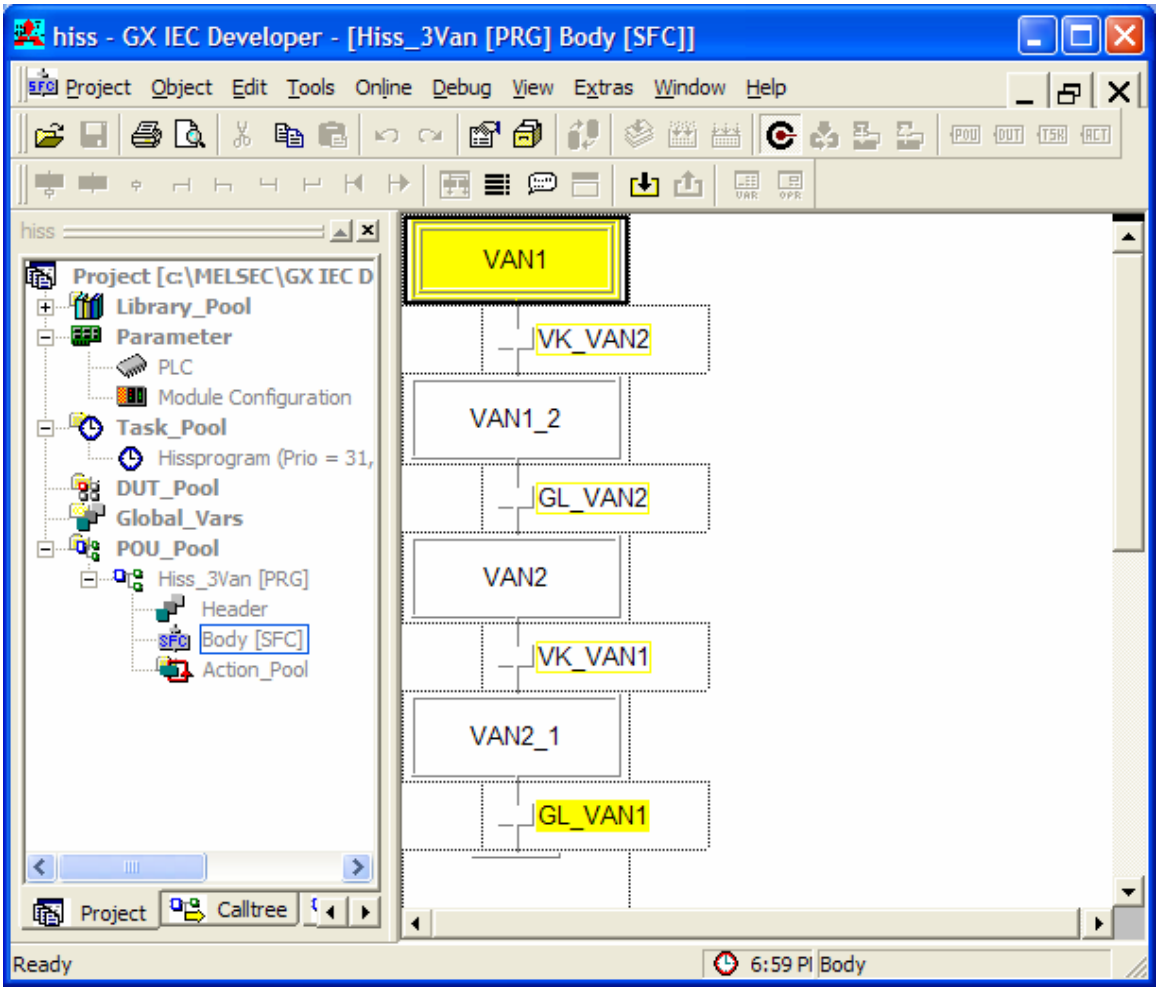
Figur 9

Nu är det dags för kompileringen, överföring till PLC samt testning.
Ha fortfarande Hissprogram markerat i **TASK_Pool**. Välj nu Project-> Rebuild All.

Om inga kompileringsfel välj i så fall överföring Project-> Transfer-> Download to PLC

Du tvingas på nytt att mjukvarustoppa PLC och återstarta densamma för att överföra programmet.
Nu när programmet är överfört skall vi testa det genom att provköra hissen, men klicka på Body[SFC] för din POU Hiss_3Van. Markera fönstret där Body:n ligger och starta monitorering (piltavlan). Då ser det ut enligt figur 10.

Ställ hissen manuellt på våning1 om den inte står där. Kör programmet !
Det gulfärgade tillståndet är det just nu aktiva och övergångsvillkoret GL_VAN1 är just nu sant.
Tryck på Våningsknapp 2 på hissmodulen. Se vad som händer i fönstret !

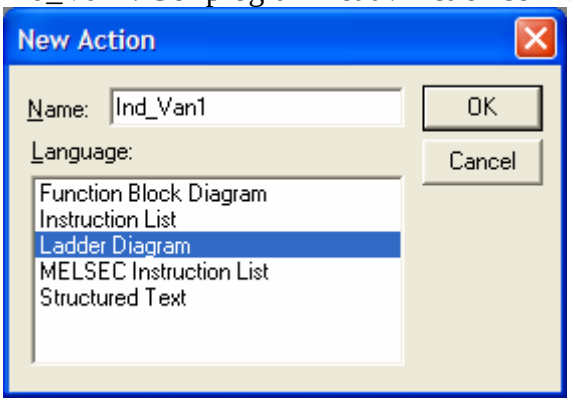


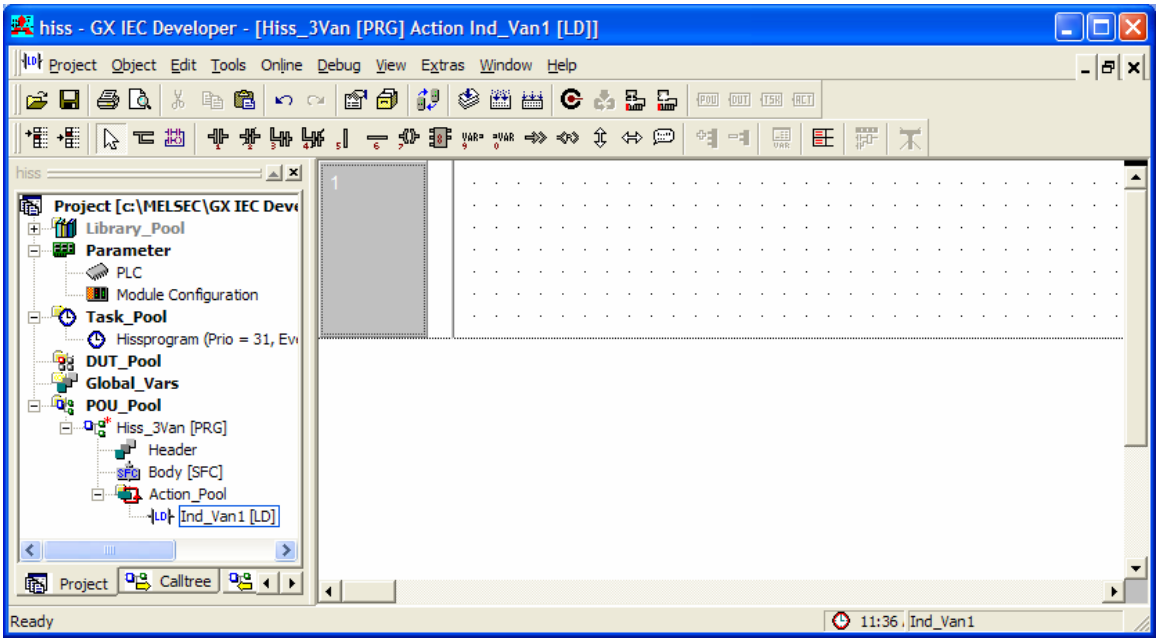
Figur10

Nu har vi kört ett väldigt enkelt PLCprogram där vi inte har använt några komplicerade Actions eller Transitions utan dessa har enbart varit globala variabler.

Fördröjning – ”Timer”

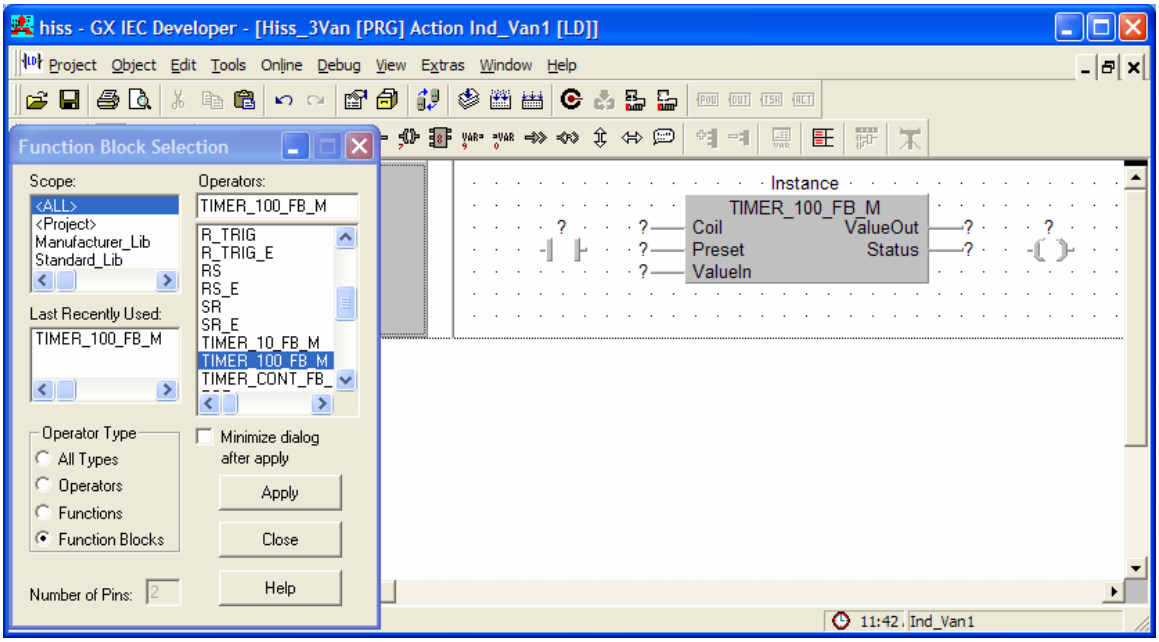
Vi skall nu fördröja olika händelser och då behöver vi ett timerelement. Eftersom detta är en Aktivitet/händelse så ska vi skapa en Action. Markera Action_Pool i navigatorn och välj New Action. Vi skall fördröj tändningen av lampa på våning 1. Kalla Action för Ind_Van1. Gör programmet av Action som ett ladderdiagram.





Figur 11

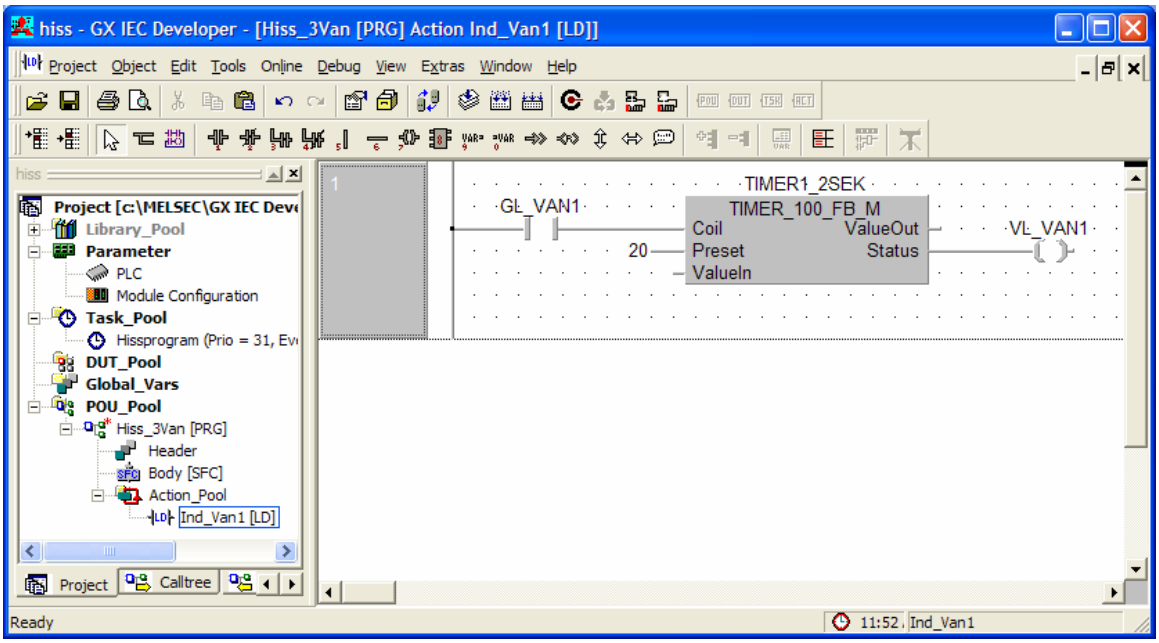
Dubbelklicka på Action: IND_VAN1 och ni får som i figur 11. Lägg pekaren i det prickade fältet och använd högerknapp på musen och lägg in en contact, coil och ett function block (FB). Det ser då ut som i figur 12.



Figur 12

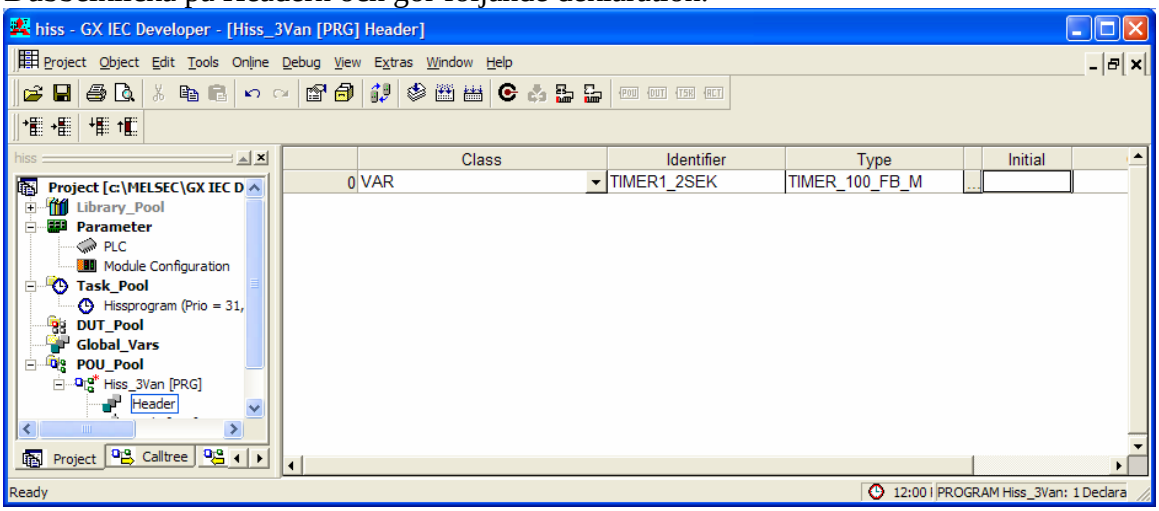
Förbind dessa och plocka bort frågetecknen framför **ValueIn** och **ValueOut**. Använd högerknapp på musen och välj **Interconnect Mode**. Använd högerknappen på musen på nytt och välj Select Mode och skriv på frågetecknen ovanför ”contact”, ”coil” respektive vid ”Preset” följande: GL_VAN1, VL_VAN1 samt 20.

Nu återstår en sak nämligen att namnge själva timer funktionsblocket. Skriv över Instance : TIMER1_2SEK. Din Action bör se ut enligt nedan i figur 12.



Figur 13

Nu har vi infört en timervariabel som inte är någon global variabel, men den måste ändå deklareras eftersom den används i vår POU så deklarerar vi den lokalt i **Headern.** Dubbelklicka på Headern och gör följande deklaration.



Figur 14

Nu måste vi bara tala om för POU:n var skall vi ha vår ”action” någonstans i programmet. Lämpligt är väl att lägga denna i tillstånd VAN1. Tag bort den gamla ”action” VL_VAN1 och lägg in IND_VAN1 istället.

Gör en liknande ”action” och fördröjning för våningslampa 2. Infoga den i ”bodyn”.

Därefter så överför vi och testar vårt nyskrivna program:
Markera task Hissprogram. Project-> Rebuild All och om inga fel
Överför till PLC med: Project-> Transfer->Download To PLC

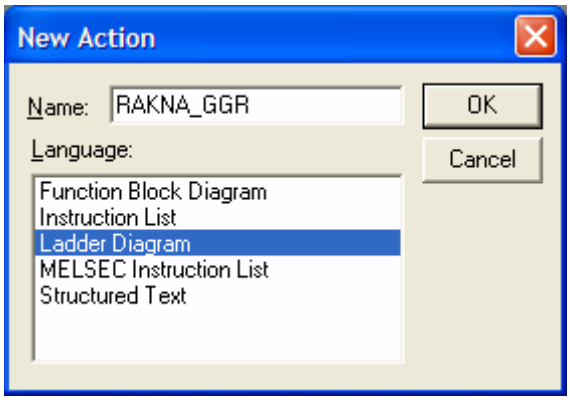
Kör hissen några gånger upp och ner för att se att det fungerar som det är tänkt.
Monitorera gärna motsvarande ”action” !
Detta görs genom att klicka i fönstret som du vill monitorera och därefter (piltavlan).

Flankräknare/Händelseräknare - ”Counter”

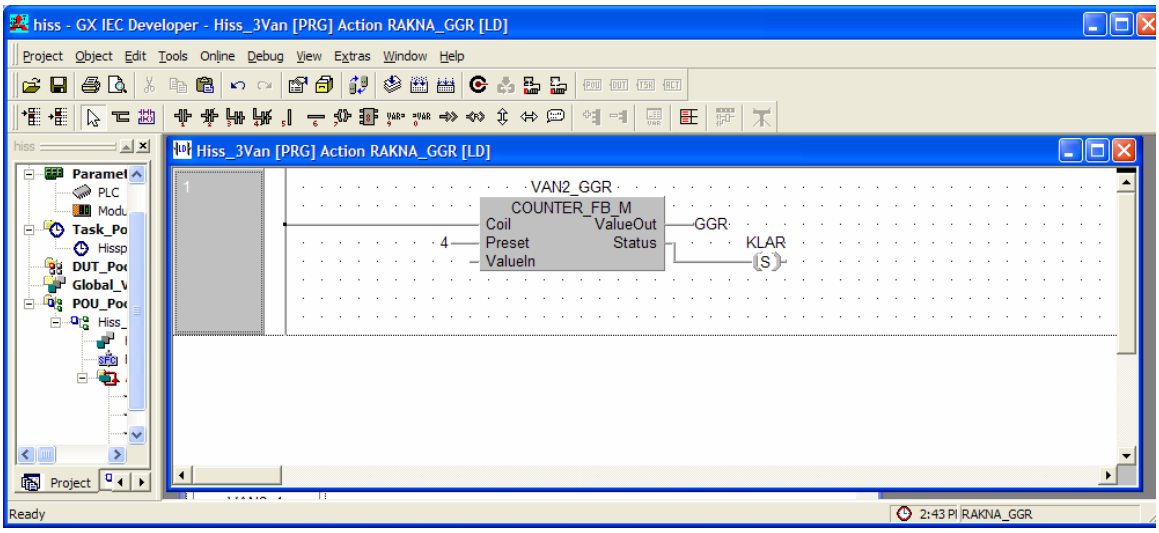
Om allt fungerar så går vi vidare för att lägga till ytterligare funktionalitet i programmet.
Nu skall vi lägga till en flankräknare/händelseräknare (Counter) inte att blanda ihop med fördröjning av tid (Timer) som vi just gjorde.

Min tanke är att vi skall räkna hur ofta vi har besökt våning 2., så vi skall skapa ytterligare en ”action” för att räkna hur ofta vi besöker våning2.

Markera Action_Pool i navigatorn och välj New Action !
Döp denna ”action” till RAKNA_GGR och programmeringsspråk Ladder Diagram.
Om inget händer när du försöker skapa en ny ”action” kan det bero på att monitoreringen inte är avstängd. I monitoreringsläge går det inte att programmera.

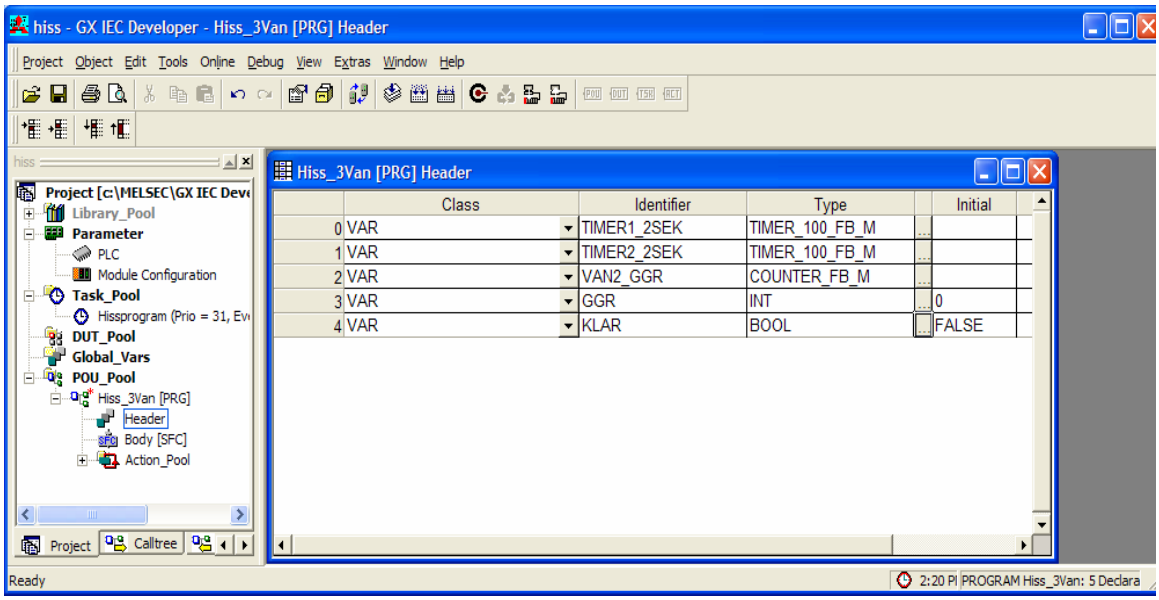


I figur 15 visas hur denna ser ut. Denna skapas på samma sätt som action med föröjning fast vi hämtar endast ett funktionsblock COUNTER_FB_M. Vi ställer in hur långt räknaren skall räkna. Sätter 1000 ggr. Variabeln GGR som finns i figuren gör att det är möjligt att se hur många gånger vi har besökt våning 2.
Notera att i coil finns ett S. Det står för ettställning av variabeln KLAR.
Denna ettställning kvarstår oavsett om vi ligger kvar i tillståndet eller om villkoret har blivit falskt.
Hur får vi fram denna. Välj en vanlig ”Coil” och dubbelklicka i denna och välj ”Set”.



Figur 15

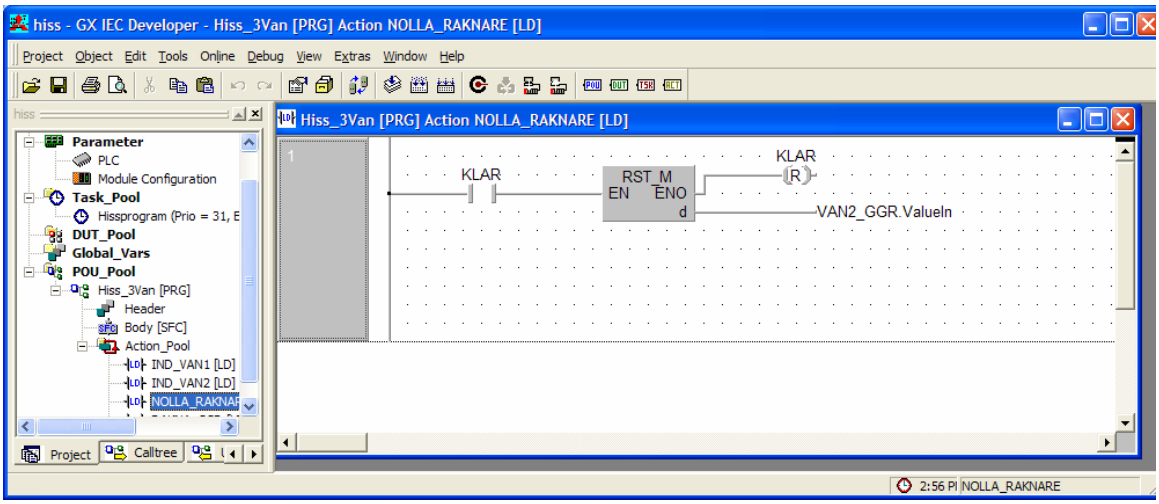
I figur 16 visas hur deklarationen ser ut. GGR är av typen INT och variabeln VAN2_GGR är av typen COUNTER_FB_M. Dessa skrivs in i Headern. De är bara lokala variabler. Variabeln KLAR är en BOOLEAN. Hur vet vi detta ? Dubbelklicka på Funktionsblocket COUNTER_FB_M så får vi upp en deklarationslista för densamma.



Figur 16

Skriv nu in din nya ”action” i POU body också. Lämpligt borde vara att lägga denna i tillståndet VAN2. Man kan ha flera ”actions” i ett och samma tillstånd. Om vi nu kompilerar och överför programmet till PLC:n, så kommer detta fungera så att om vi besöker våning2 4gånger med hissen så upphör uppräkningen. Vi har satt denna till 4 och då räknar inte VAN2_GGR längre. Vi skulle behöva komplettera vår

COUNTER_FB_M med en nollställning. Detta är en god regel. Vi behöver en action som nollställer räknarvariabeln VAN2_GGR.
Vi inför en ny ”action” som heter NOLLA_RAKNARE. Den visas i figur 17.
Notera att funktionen RST_M är en funktion inte ett funktionsblock när du skall hämta denna. Funktionen är inte någon variabel så deklaration av denna är onödig.



Figur 17

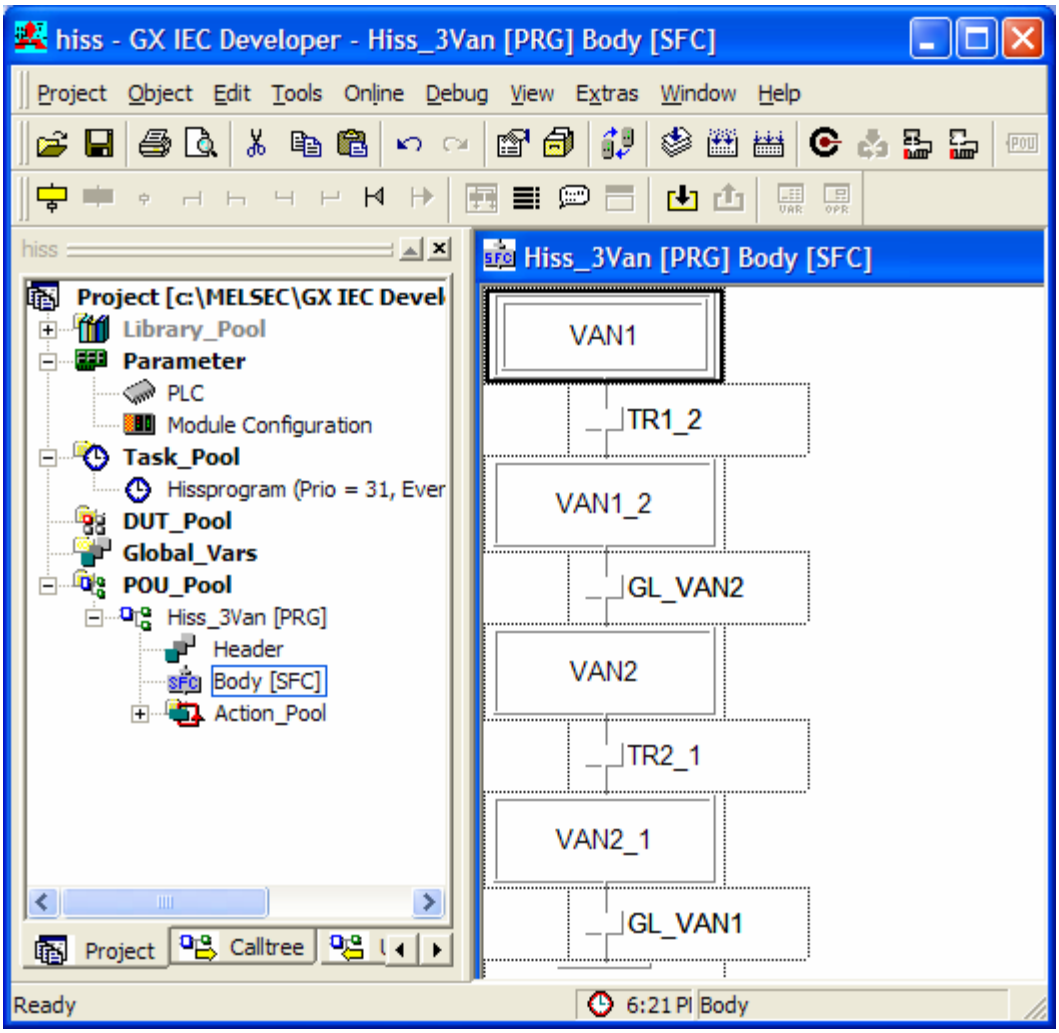
Var skall vi lägga in denna ”action” ? Tja jag tycker att i tillstånd VAN1 borde vara lämpligt. Skriv in NOLLA_RAKNARE under den befintliga ”action” IND_VAN1.

Kompilera programmet och överför till PLC. Testa och monitorera speciellt ”action” som innehåller COUNTER_FB_M. Kontrollera att nollställningen är ok!

Övergångsvillkor – ”Transition”

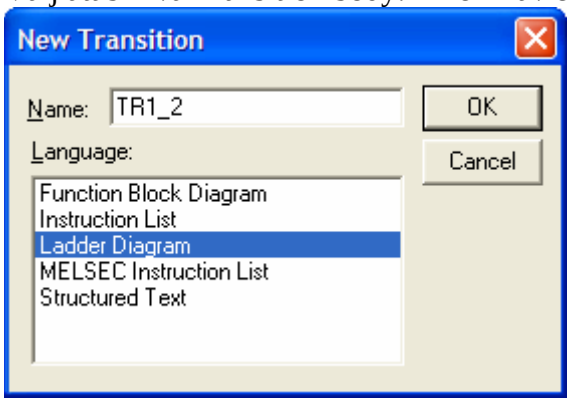
Hittills har våra övergångsvillkor mellan tillstånden (”Steps”) enbart varit en global variabel, men givetvis kan villkoret vara mycket mera komplext och då krävs att man gör en ”Transition body”.
Tänk i vårt fall med hissen att den kan styras antingen av en våningsknapp eller en hissknapp (horisontellt liggande), d v s vi skulle behöva ett eller-villkor mellan tillstånden. Vi skall införa ett eller-villkor både när vi åker upp och när vi åker ner.
Både HK_VAN2 eller VK_VAN2 skall kunna användas för att åka upp till våning2 och HK_VAN1 eller VK_VAN1 kan användas för att åka ner från våning 2.

Skriv över 2 av de tidigare övergångsvillkoren i POU-body:n enligt figur nedan !



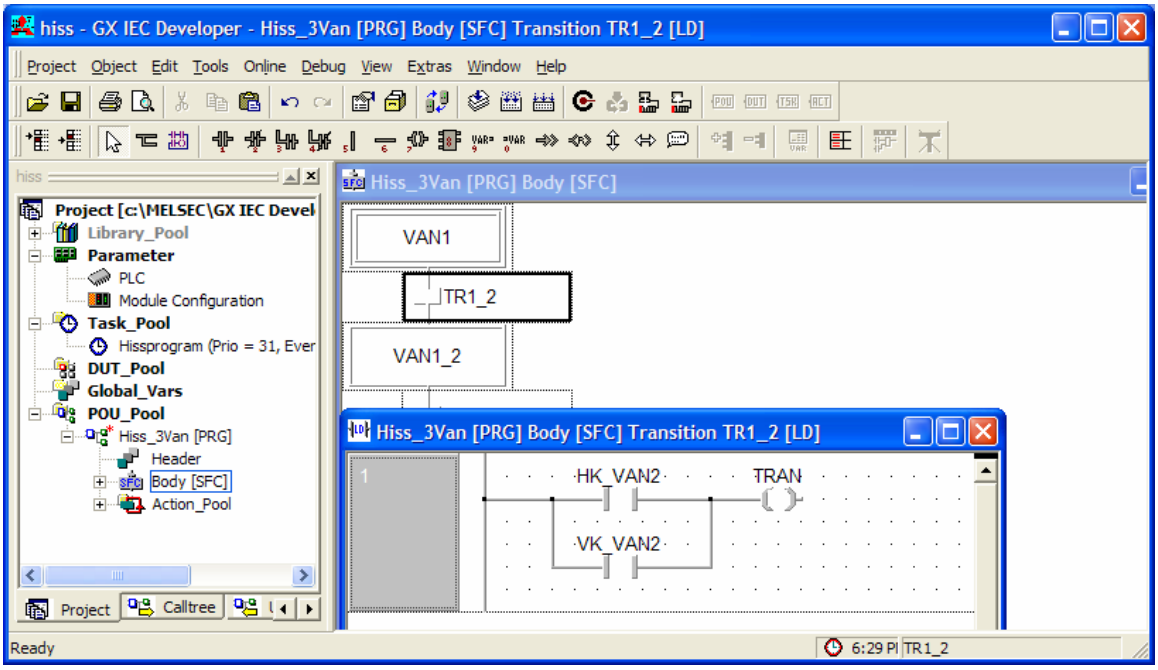
Figur 18

Därefter klickar ni på fyrkanten bredvid namnet på det nya övergångsvillkoret.
Välj att skriva Transition body:n i form av ett Ladder diagram.



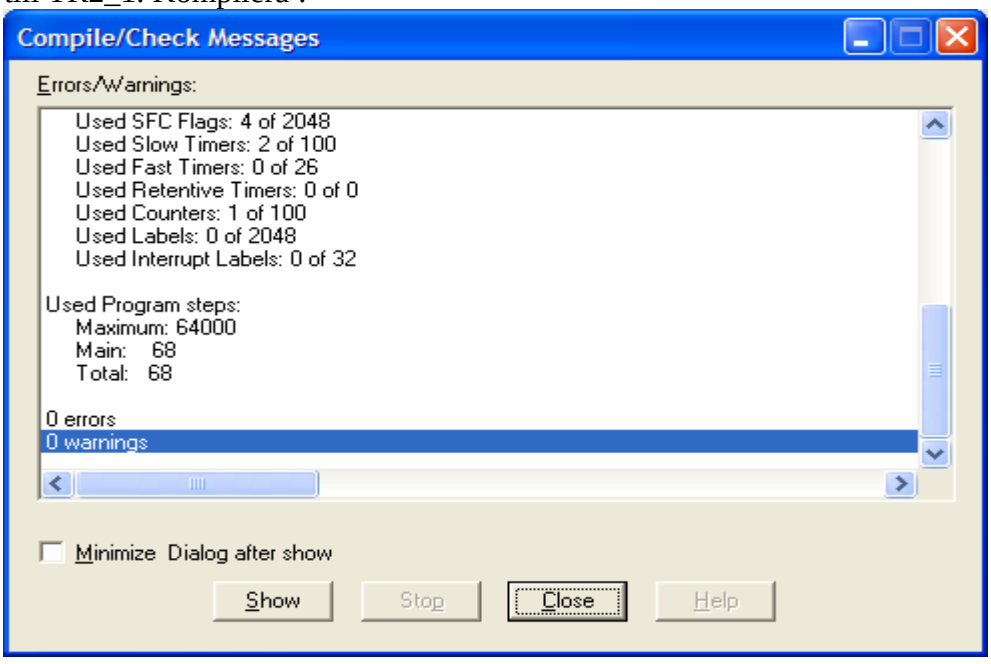
Övergångsvillkoret mellan tillstånd VAN1 och VAN1_2 döps om till TR1_2.
Se figur 20! Den består endast av två parallellkopplade kontakter (eller-villkor) kopplat till en ”coil” som heter TRAN. TRAN är ingen variabel i egentlig mening utan ett

fördefinierat ord i programvaran som helt enkelt står för TRANSITION. Denna behöver inte deklareras. Om villkoret för TRAN är uppfyllt. Ja då är övergångsvillkoret sant och vi lämnar tillstånd VAN1 och går till tillstånd VAN1_2.



Figur 20

Vi gör ett liknande övergångsvillkor mellan tillstånd VAN2 och VAN2_1. Denna döps till TR2_1. Kompilera !



Notera att vid kompileringen står det hur många Counters, Timers och tillstånd som du använder. Fortsätt med överföring och testning av programmet.

Fungerar det att använda olika knappar för att åka upp och ner ?

Inlämningsuppgift: din uppgift är nu att skriva ihop ett PLC-program för styrning av en fyrvåningshiss. Hissen styrs antingen av hissknappar eller våningsknappar.
Vid en beställning skall våningslampan blinka ända tills hissen har nått dit.
Flera samtidiga beställningar kan göras av hissen, men exekvera dessa i tur och ordning, men tänk på att när hissen har börjat röra sig i en riktning skall den fortsätta åt samma håll tills alla beställningar är tagna. Den skall stanna på varje beställd våning i 2 sekunder innan den åker vidare.
Använd 4 räknare (counter) för att hålla reda på hur många gånger som hissen har stannat till på de olika våningarna.
PLC-programmet måste testas och demonstreras mot vår Hissmodul för att bli godkänt..

Ett demoprogram av GX IEC Developer (ver 7.01) finns på Beijers hemsida:

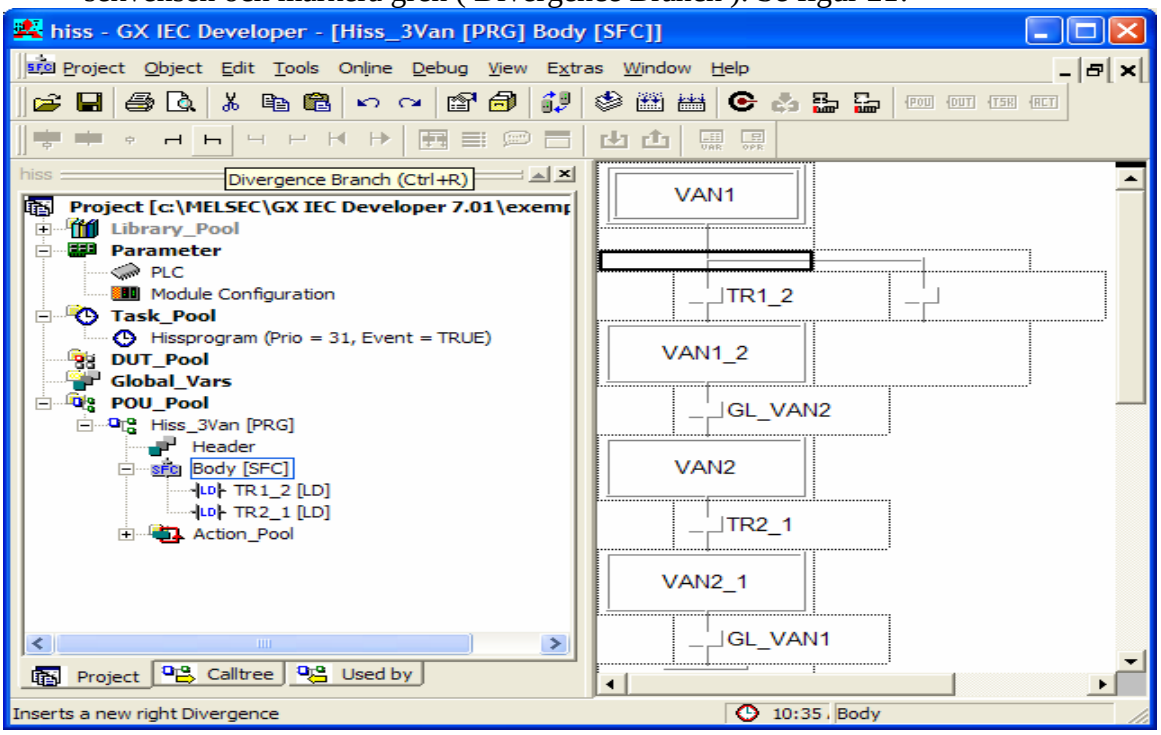
http://www.beijer.se/web/web_aut_se.nsf/AllDocuments/BAE711C83A36EE3CC1256FAC002871CE

Denna kan ni använda för att ladda ner programvara och skriva/förbereda labuppgifter hemma. Programvaran skall förmodligen också ligga på era projektrumsdatorer.

BILAGA

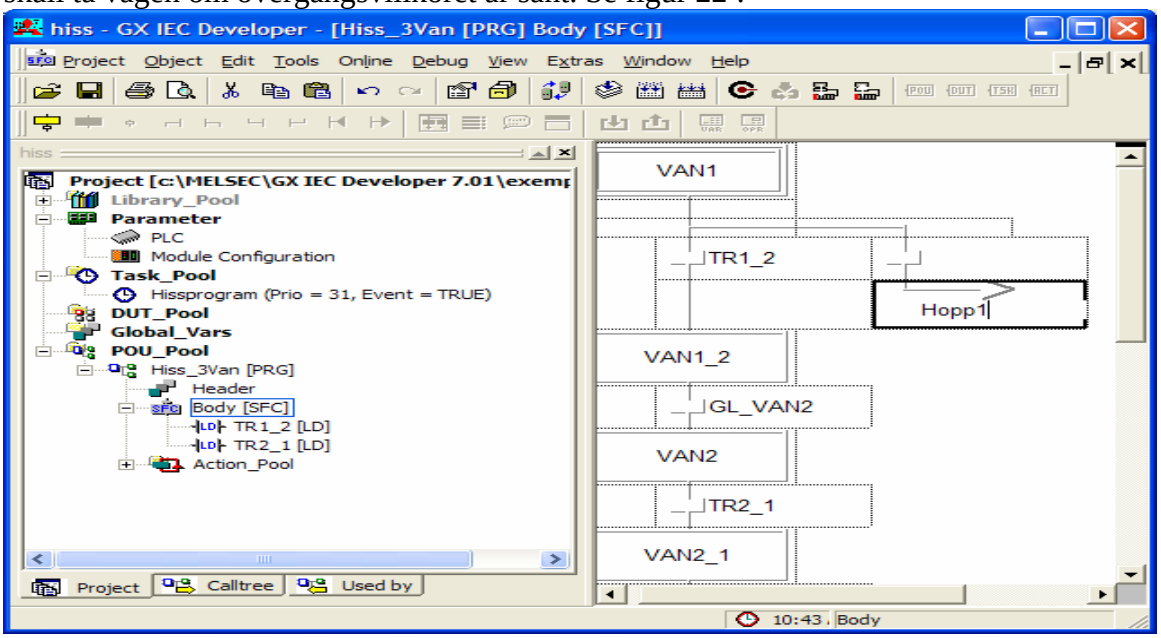
Tips till laboration 2 och inlämningsuppgift.

Kanske vill du skapa alternativgrenar respektive parallellgrenar eller programhopp. Detta kan göras genom att du markerar övergångsvillkor där du vill grena sekvensen och markera gren (Divergence Branch). Se figur 21!



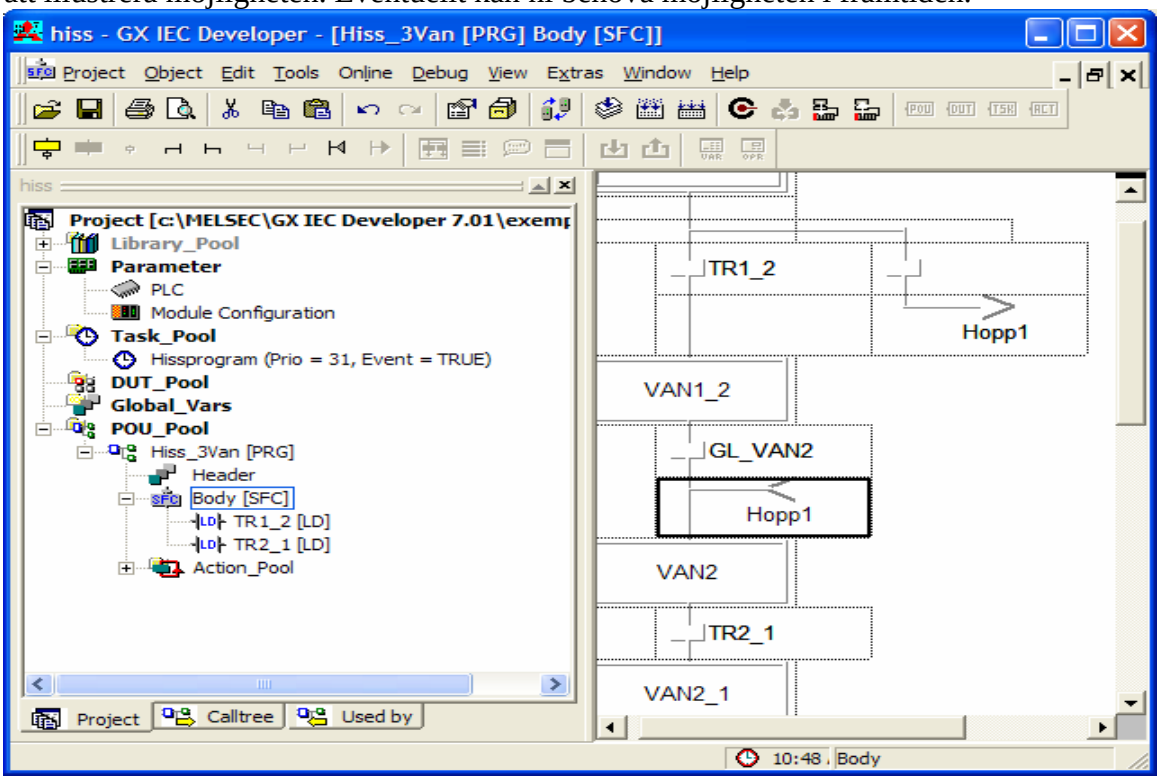
Figur 21

Man kan också lägga till ett hopp efter övergångsvillkoret istället för att åstadkomma en ny sekvensgren. Du sätter en etikett (Label) på denna för att tala om var programmet skall ta vägen om övergångsvillkoret är sant. Se figur 22 !



Figur 22

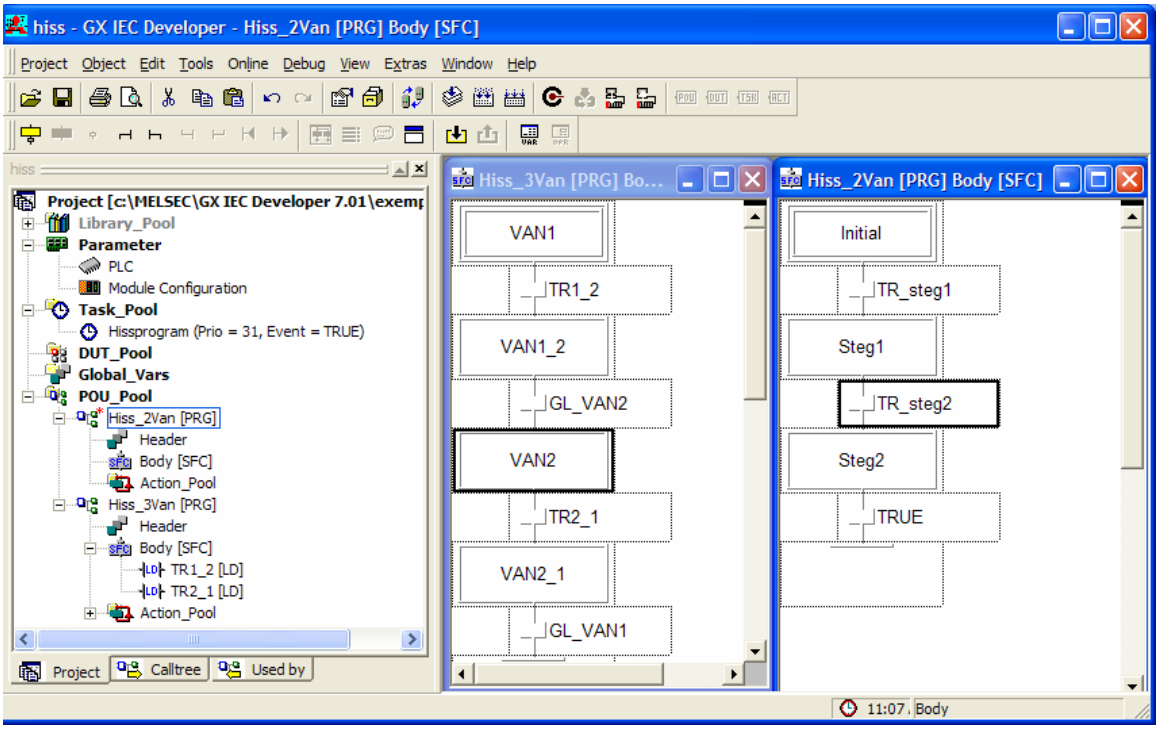
Det måste alltså finnas ett inhopp någonstans med samma etikett på denna.
Se figur 23 nedan ! Programmet nedan har ingen nytta av detta hopp utan finns enbart för att illustrera möjligheten. Eventuellt kan ni behöva möjligheten i framtiden.



Figur 23

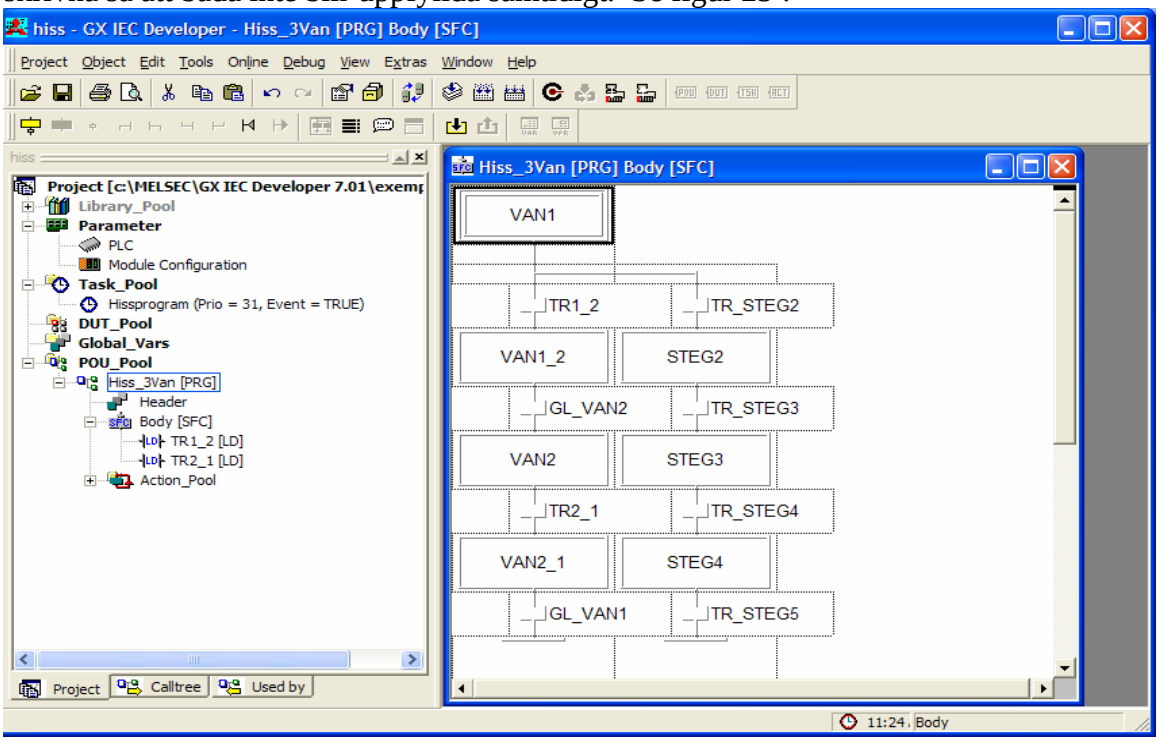
I ert styrprogram kan ni även ha flera sekvenser som utförs samtidigt. Båda sekvenserna aktiveras samtidigt och dess initialtillstånd blir aktivt. Därefter hur fort eller sakta den ena eller andra sekvensen utförs hänger på övergångsvillkoren, så att en sekvens kanske stoppar den andra för att den hindrar något övergångsvillkor för att bli sant. Hur hänger dessa sekvenser ihop (olika POU:s) ?
Vi kan t ex använda en global variabel för att överföra information mellan sekvenserna.
Se figur 24 !

Det kan vara att i den vänstra POU:n, i tillstånd VAN2 så ettställs kanske en global variabel som är gemensam med den högra POU:n och som används i övergångsvillkor TR_steg2 och denna kan inte bli uppfylld om inte den vänstra POU:n har kommit ner till tillstånd VAN2 först, så genom att göra variabler gemensamma kan vi hindra en eller flera POU:s att bli utförda. Givetvis finns möjlighet att använda flera POU i ett och samma program.



Figur 24

Visar även hur en alternativgren ser ut. Vi låter sekvensen förgrena sig under tillståndet VAN1. Det är antingen den vänstra eller högra grenen som är möjlig att vandra inte bägge samtidigt, men givetvis måste övergångsvillkoren TR1_2 och TR_STEG2 vara skrivna så att båda inte blir uppfyllda samtidigt. Se figur 25 !



Figur 25

Figur 26: Fyra-våningshiss