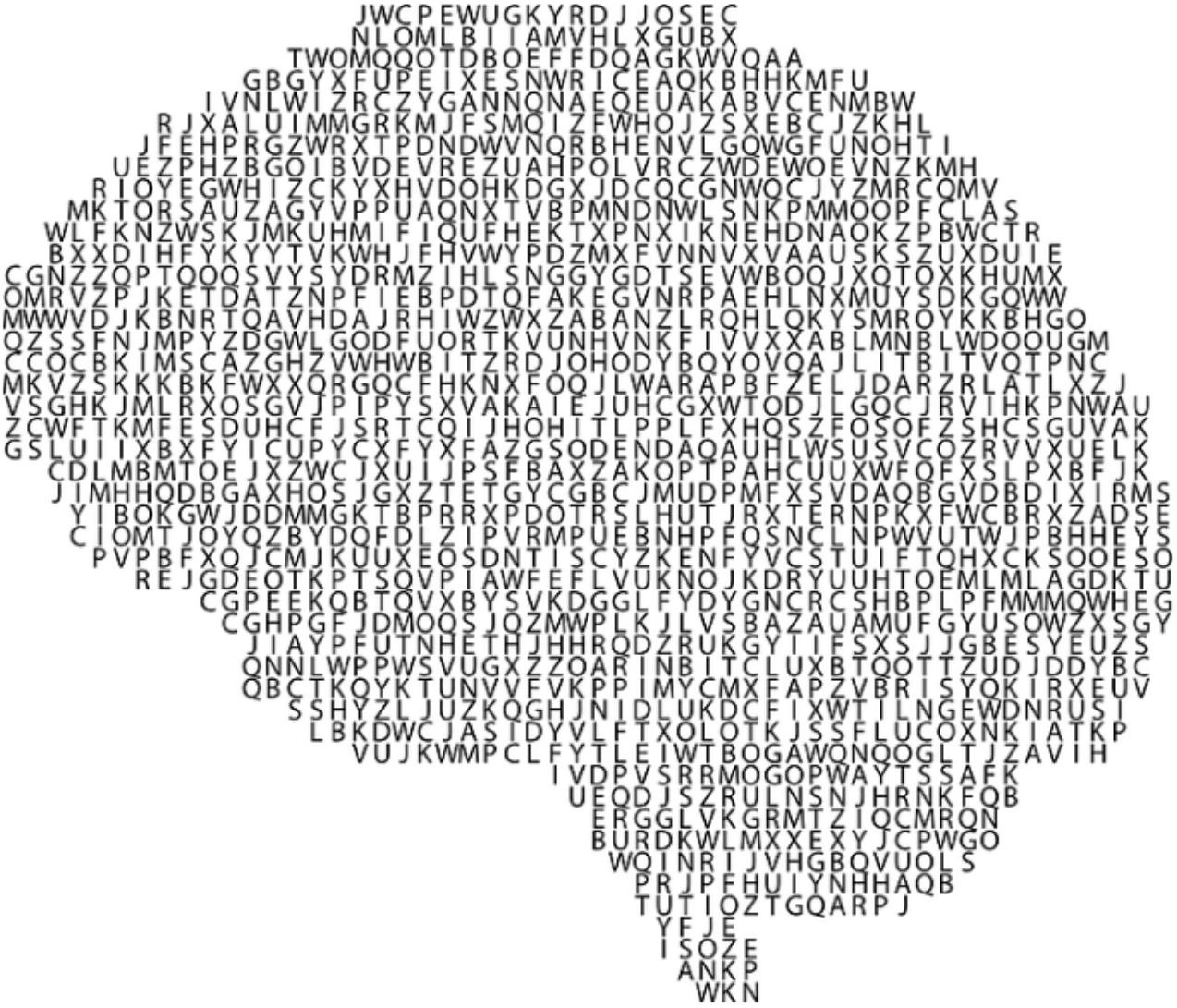
Large Language Model (LLM) og generering af Ladder Diagram (LAD)



(Dezhic, 2023)

Indhold

Abstract

This report explores the application of large language models (LLM), specifically ChatGPT-4, in the development of ladder logic programming (LAD) for Siemens Programmable Logic Controller (PLC). The project identifies and evaluates four different methods for generating LAD code, focusing on action research and iterative prompt development. The most effective approach is highlighted as a strategy where accurate prompts, and relevant examples enable the generation of a fully functional motor function block with an integrated counter function, ready for import into Siemens program development tool Totally Integrated Automation Portal (TIA Portal).

The study identifies challenges in achieving direct integration between ChatGPT and the TIA Portal, primarily due to syntactic limitations and the current capabilities of the model. Additionally, strategies such as Chain-of-Thought prompting and Few-shot learning are discussed, enabling structured and dialog-based development processes.

Finally, the report contrasts the use of pre-tuned and fine-tuned models, emphasizing how fine-tuning could address more complex tasks where pre-tuned models face limitations.

Forord

Intentionen med denne rapport er at belyse, hvordan LLM kan anvendes praksis i automationsbranchen, med særlig fokus på LAD. Målet er at undersøge forskellige prompts og videreudvikle dem undervejs som en del af aktionsforskning. Denne rapport vil belyse fire konkrete metoder til LAD generering samt undersøge prompt.

I dette projekt anvendes ChatGPT-4-modellen til at generere LAD kode. Rapporten er derfor afgrænset til denne specifikke model, men andre modeller er også blevet undersøgt og diskuteres i rapporten. Der eksperimenteres med forskellige metoder til opbygning af prompts, hvilket også diskuteres. Prompten er designet til at fungere med ChatGPT-4, en pre-tuned model, med det formål at undersøge, om modellen kan generere tekstbaserede lavprogrammeringssprog, der efterfølgende kan konverteres til LAD, som modellen ikke er trænet specifikt i.

På baggrund af undersøgelsen vil der blive analyseret de 4 metoderne og hvad deres nuværende potentiale er samt fremtidigt potentiale. Rapporten vil blive offentliggjort på eaviden.dk.

Indledning

Dette projekt fokuserer på brugen af LLM til generering af kode, med særligt fokus på anvendelsen af ChatGPT4 til udvikling af LAD kode logik til Siemens PLC'er. Fire forskellige metoder til generering af LAD kode med en LLM bliver undersøgt, hvoraf én metode har vist sig særligt effektiv, når den understøttes af korrekte prompts og relevante eksempler. Denne metode demonstrerer, hvordan en motorfunktionsblok, baseret på eksisterende data, uploadet til modellen, kan udvide logikken med en tællerfunktion for antal af start og genereres som en færdig funktionsblok i Statement List (STL) der kan konverteres til LAD, der efterfølgende kan importeres direkte i TIA portalen.

En anden metode viser et stort potentiale for at skabe en mulig direkte integration mellem ChatGPT og TIA portalen. Dog afslører projektets undersøgelser, at de nuværende syntakser og funktioner i værktøjerne gør denne tilgang udfordrende, og i praksis næsten umuligt.

Projektet udforsker også, hvordan man bedst kan strukturere prompts til en pre-tuned model, især når programkoden kræver specifikke syntakser, som modellen ikke er trænet på. Her anvendes strategier som Chain-of-Thought (CoT) prompting og Few-shot learning i forskellige udviklingsfaser. Opgaverne brydes ned i mindre, håndterbare dele, hvilket gør det muligt at føre en dialogbaseret samtale med ChatGPT under og efter hver fase af kodegenereringen.

Endelig perspektiverer projektet forskellen mellem at anvende en pre-tuned model, som i dette projekt, og en fine-tuned model. Det diskuteres, hvordan fine-tuning kan åbne op for løsninger af mere komplekse opgaver, hvor pre-tuned modeller har sine begrænsninger. Der bliver også perspektiveret til den samfundsmæssige betydning af at anvende en LLM.

Problemformulering

Der er sket en stor fremgang indenfor LLM algoritmer, og der kan opstå nye muligheder for at genere PLC kode. Kan disse modeller hjælpe med at genere kode, hvilke programsprog, er relevant at undersøge nærmere og hvordan får man en LLM til at genere kode. En industriel PLC som Siemens understøtter mange forskellige programsprog, så som Structured Control Language (SCL), Statement List (STL), Ladder Diagram (LAD) og Sequential Function Chart (GRAPH). Det kommer an på programmørens fortrukne sprog og PLC type. LLM er gode til tekstprogrammerings sprog, men hvad hvis man vil have den til at genere LAD kode. Et eksempel kunne være at en slutbruger eller kunde forlanger det. Det er ikke altid hensigtsmæssigt at lave alt i LAD, men hvilken metode kunne være hensigtsmæssigt, hvis man vil udnytte en LLM til at genere LAD kode.

Hypotese: LLM kan anvendes til at genere program kode i LAD til anvendelse indenfor industriel automation og dermed øge produktiviteten for virksomhederne i branchen.

Hovedspørgsmål

Hvordan får man ChatGPT til at genere LAD program som kan bruges i en Siemens PLC?

Underspørgsmål

Har andre internationale institutioner lavet lignede undersøgelser, hvilke erfaringer kan der bygges videre på?

Hvordan skal en prompt opbygges således man får et optimal output?

Hvilke metoder kan anvendes til at genere LAD program?

Begreber

1. Generelle teknologiske begreber

LLM (Large Language Model)
En type kunstig intelligens, der er trænet på store tekstmængder og kan generere tekst, besvare spørgsmål og udføre avancerede opgaver, som f.eks. kodegenerering.

ChatGPT
En AI-model udviklet af OpenAI, der bruges i dette projekt til at generere kode, forbedre produktivitet og hjælpe med dokumentation.

Few-Shot Learning
En prompt-teknik, hvor modellen gives få eksempler på ønsket output for at forbedre dens præstation.

CoT (Chain-of-Thought Prompting)
En metode, der opfordrer AI til at tænke trin-for-trin for at generere mere strukturerede og logiske outputs.

XML (Extensible Markup Language)
Et tekstbaseret format brugt til at opbevare og udveksle data, herunder eksport og import af kodeblokke i TIA Portal.

2. Programmeringssprog

LAD (Ladder Diagram)
Et grafisk programmeringssprog, der efterligner elektriske kredsløbsdiagrammer og bruges til at programmere PLC'er.

STL (Statement List)
Et tekstbaseret programmeringssprog for Siemens 300 og 400 serie PLC'er, som giver lavniveau kontrol og kan konverteres til LAD under visse betingelser. STL er IL efter standarden IEC 61131-3

AWL (Anweisungsliste)
Det tyske udtryk for STL-programmering, anvendt i Siemens miljøet, når der skal generes source filer

SCL (Structured Control Language)
Et tekstbaseret programmeringssprog brugt i Siemens 1200 og 1500 serie PLC'er, særligt velegnet til komplekse opgaver. SCL er ST efter standarden IEC 61131-3

GRAPH (SFC - Sequential Function Chart)

Et grafisk programmeringssprog der efterligner GRAFCET opbygning til at lave sekventiel programmering.

UDT (User-Defined Type)
En brugerdefineret datatype, som strukturerer og organiserer data i PLC-programmer.

FB (Funktions Blok)
En FB er en genanvendelig programblok, der udfører en specifik funktion. Blokken kan have et interface med input, output og data gemt i blokken for at håndtere logikken. Det kan f.eks være en motor eller en ventil.

3. Siemens-miljø

PLC (Programmable Logic Controller)
En industriel CPU designet til at styre maskiner og processer i realtid.

TIA Portal
Totally Integrated Automation portal. Siemens' integrerede miljø til at konfigurere, programmere og administrere PLC-systemer og HMI'er.

TIA Openness
En API fra Siemens, der gør det muligt at manipulere TIA-projekt via eksterne programmeringssprog som C#.

HMI (Human Machine Interface)
En brugergrænseflade, der muliggør overvågning og kontrol af industrielle processer.

Teori og metode afsnit

Forskningsprojektet benytter sig af metoden ”aktionsforskning”. Det inkluderer at man som forsker er aktivt involveret i processen. Aktionsforsknings metoden, kommer særlig til udtryk når man som forsker udvikler og afprøve forskellige prompts og vurdere outputtet af LLM, som en iterativ proces.

Prompt, der er udviklet til projektet, er baseret på relevante forskningsartikler til struktur og forlag til at udarbejde en prompt der giver et optimal output. Prompten bliver også baseret på erfaring der opstår i processen kombineret med den nødvendige viden indenfor industriel automation og programmering af STL og LAD.

LLM er kontekstafhængig, da det afhænger af det trænet data, LLM-model der bruges, samt inputtet til modellen. Derfor kan det argumenteres at den bedste metode til forskningen er aktionsforskning da prompt og data bliver udviklet undervejs i projektet, og er derfor en iterativ proces der forsøger at få et optimal output.

Til indsamling af data til analysen vil jeg gøre brug af følgende metoder:

Informationssøgning: Formålet er at undersøge relevante projekter, der anvender LLM inden for industriel automation, med henblik på at bygge videre på eksisterende løsninger eller vurdere, om de kan besvare problemstillingen. Derudover søges der inspiration til udvikling af promptmetoder og forslag til at opnå optimalt output fra en LLM-model.

Eksperimentere: Udvikle forskellige prompt og metoder til at genere programkode til LAD programsprog.

Afgrænsning:

Projektet vil udelukkende arbejde med OpenAI LLM-modeller, såsom ChatGPT-4.

Fokus for projektet er på genereringsmetoder og udvikling af prompts til et specifikt programmeringssprog og en PLC-type, herunder LAD og Siemens PLC. TIA-programmeringsværktøjet vil blive anvendt til at undersøge og opbygge mulige syntakser samt eksempler på konkrete funktioner. Desuden vil flere af TIAs værktøjer blive undersøgt i projektet.

Informationssøgning

I dette afsnit redegøres der for metoden til informationssøgning, henvisning til relevante artikler og forklare deres anvendelse i projektet.

Der er foretaget søgninger på forskellige hjemmesider for at finde artikler inden for Generativ AI, anvendt i PLC-programmering. De primære ressourcer, der er anvendt, er eaviden.dk og Google Scholar. Her er der brugt søgeord som "LLM Industrial Programmering", "LLM Industrial Automation" og "AI Industrial Automation". Da det er en relativt ny teknologi, er der ikke mange artikler, der specifikt handler om LLM og Industriel Automation programmering. Den generelle søgning startede i foråret 2023

De fleste artikler, omhandlede emner som billedgenkendelse og kvalitetskontrol ved hjælp af AI. Derudover var der også en del artikler, der fokuserede på forebyggende vedligeholdelse ved at overvåge systemer i anlæg. Her kan AI anvendes til at overvåge parametre som vibrationer og temperatur for at forebygge nedetid på systemer og planlægge vedligeholdelsesarbejde.

Dog var der en relevant artikel ”ChatGPT for PLC/DCS Control Logic Generation” fra Heiko Koziolek, ABB Research, Ladenburg, Germany.

Artiklen præsenterer en samling af 100 prompts, der tester en LLM's evne til at generere korrekt og nyttig kontrollogik (Koziolek, 2023). Disse prompts er struktureret systematisk og dækker forskellige aspekter af industrier og program metoder.

Artiklen diskuterer brugen af ChatGPT til at nedbryde kode i adskillige metoder med konkrete funktioner, som i tilfældet med 'ELEVATOR CONTROL'. Det blev forklaret, at den kunne løse logikken, men den havde endnu ikke implementeret en optimeret planlægning for flere samtidige anmodninger eller håndtering af nødsituationer. Den genererede kode kan raffineres og udvides i opfølgende prompts. Dette viser, hvordan LLMs kan bruges til at generere og forbedre kode, hvilket er særligt relevant for projektet.

Artiklen fremhæver, at der gennem mange år er udviklet metoder til at automatisere genereringen af logikkode, da udviklingen af sikker og effektiv kontrollogik er omkostningskrævende. Over de sidste to årtier har både forskere og praktikere arbejdet på at udvikle metoder, der kan generere kontrollogik ud fra forskellige elementer og modeller. Der er blevet anvendt forskellige metoder, herunder regelbaserede tilgange, der genererer kontrollogik fra Piping and Instrumentation Diagram (P&ID), og modeldrevne paradigmer, der genererer kontrollogik fra Unified Modeling Language (UML) eller Systems Modeling Language (SysML). Praktikere har også skabt skabeloner til kontrollogik-kode ved at importere information fra tabelbaserede I/O-lister ind i programmeringsmiljøer og derefter manuelt tilføje det, der kan betegnes som "lim-logik". Nogle har endda kodet dele af kontrollogikken i systemkontroldiagrammer (SCD) for at opnå en direkte oversættelse til IEC 61131-3 standard.

Artiklen diskuterer også udviklingen LLM og deres indvirkning på kodegenerering. Den nævner, at med den udvikling der sker indenfor LLM, har nogle spekuleret i at nogle programmeringsopgaver og derved jobs kan blive erstattet af LLM. For eksempel opnåede DeepMinds specielt trænede AlphaCode høje ratings, når den løste opgaver fra officielle programmeringskonkurrencer. GitHub Copilot er blevet populær til at generere standardkode eller til at optimere eksisterende kode. Dog blev ingen af disse tilgange analyseret for deres kapabiliteter i forhold til typiske kontrolteknikopgaver.

Artiklen diskuterer også, hvordan ChatGPT kan bidrage med domænespecifik viden. Programmøren står ofte over for processer, som kræver stor indsigt i processen for at forstå hvordan anlægget eller processen skal reagere i driften og ved fejlsituationer. Derfor kræver det mange samtaler med procesingeniører og at de

har tid til at lære programmøren systemet og processen. GPT-4's træningssæt indeholder store mængder af produktionsdomæneviden, som kan hentes med de rigtige prompts. ChatGPT kan således give præcise svar på specifikke spørgsmål, uden at programmøren behøver at læse lange instruktionstekster eller produktionshåndbøger. ChatGPT har fysiske og almindelige ræsonnementsevner, som kan fremskynde løsningen af virkelige automatiseringsproblemer. Dens trænede data viser en forståelse af et trafikstyringslys eller en turbine i et kraftværk, selvom ChatGPT ikke inkluderer en eksplicit kodet 'verdensmodel'. Neurale netværk i GPT-4 kan syntetisere nyttige tekst mønstre baseret på statistikker omkring disse situationer. Udover produktionsdomæneviden indikerede ChatGPT også dyb teknisk viden om kommunikationsprotokoller eller specifikke aspekter af ST-sproget. Grænserne for dens viden er at den kun er trænet på offentlig viden og ikke konkrete løsninger fra leverandører, men de fandt ud af, at ChatGPT allerede kan tackle mange situationer uden brugerdefineret træning.

Det er derfor også interessant for dette projekt da de argumenterer for at den har en masse viden og domænespecifik viden og det er derfor ikke nødvendigt at træne den på brugerdefineret data i mange situationer. Dog kan mange processer i industriel automation være komplekse og kræver derfor ofte domænespecifik viden for at sikre korrekt styring af systemet.

Artiklen bemærker, at svar fra ChatGPT kræver omhyggelig validering. Mange gange ser ChatGPT-svar korrekte og velformulerede ud, men kan også indeholde subtile fejl eller udeladelser. Syntaktiske problemer kan let identificeres ved at forsøge at kompilere den genererede kode. Dog bemærker ChatGPT ofte, at de genererede svar skal valideres af en ekspert, før de implementeres i en rigtig applikation. Dette understreger vigtigheden af at have domænespecifik viden indenfor PLC programmering, når man arbejder med LLMs som ChatGPT.

Forfatterne af artiklen gennemførte deres undersøgelse ved hjælp af ChatGPT4 modellen, på grund af dens udbredte tilgængelighed og gode brugervenlighed. De valgte at generere hovedsageligt IEC 61131-3 Structured Text (ST) programmer, som er et af de mest populære tekstbaserede kontrolprogrammeringssprog, der understøttes af mange PLC og DCS udviklingsmiljøer. Selvom artiklen bruger IEC 61131-3 Structured Text (ST) til deres eksperimenter, anerkender de populariteten af ladder-programmering (LAD). De bemærker dog, at LAD er svære at generere på grund af dens grafiske natur. Selvom det er tænkeligt, at den genererede ST-kode kunne omdannes til LAD, har de ikke direkte eksperimenteret med dette i deres undersøgelse.

Det er noget som dette projekt vil undersøge, hvordan kan man genere LAD med chatGPT4.

Samlet set har denne artikel været en værdifuld kilde til information for projektet. Den har givet indsigt i, hvordan LLMs kan bruges til at generere kontrollogik, og har præsenteret en række eksempler og case-studier, der kan være nyttige.

Prompting litteratursøgning:

Under eksperimenterne blev det tydeligt, at modellen havde svært ved at generere output, der ikke var for generiske og samtidig anvendelige i kodningen. For at adressere denne udfordring blev begrebet "Prompting" undersøgt, og forskellige metoder blev implementeret, herunder "chain-of-thought (CoT) prompting" (Zhang, 2022). Denne metode anvender en simpel prompt som "let's think step-by-step" for at forbedre kvaliteten af outputtet.

Ethan Mollick følger udviklingen af LLM og generelt omkring prompting. Her undersøger han og læser relevante artikler, som beskriver måder at prompte på, og frigiver nyhedsbreve på området, hvad er nyt og relevant. Disse nyhedsbreve har også været en inspirationskilde til nogle promptmetoder, der florerer på nettet og i forskningsverden. Han undersøger relevante artikler og formidler gode og dårlige resultater, så som ”Take a depth breath” (Google DeepMind, 2024) promptmetoden, som har været en interessant metode og anerkendt i et stykke tid, da denne metode forsøger at få den til at agere med følelser, nogle har endda forsøgt at true modellen (Mollick, oneusefulthing, 2024) så som at den vil få bøder og få COVID, men også forsøgt med positive følelser, så som gaver til Taylor swift koncert og verdens fred og fået gode resultater ud af det. Dog har andre afprøvet disse prompting metoder og fundet dem usikker eller ikke har kunne give samme resultater og var derfor en meget situation bestemt og stor usikkerhed i resultaterne. Derfor skriver Ethan, at man ikke behøver at prøver sig frem med disse ”magiske” metoder. Dog fremhæver han 3 metoder, som han mener er relevant ift. prompting og som er blevet benyttet i projektet:

- kontekst
- Few shot (eksempler)
- CoT

Man skal give *kontekst,* så som hvordan skal den opføre sig, hvad er dens opgave og hvad skal den give af svar. Man skal også give den *eksempler,* så som hvordan koden skal se ud, hvilke syntakser kan anvendes og i hvilken kombination, samt give eksempler på gode og dårlige output. Man skal anvende ”lets *think step by step*” i prompten for at løse en opgave bedre, da modellen i sig selv bliver mere logisk og struktureret i sin tilgang til at løse problemet.

Denne litteratursøgning har hjulpet med at få opbygget eksempler, prompt og forståelse af vigtigheden af god prompting for at få mere ud af en pre-tunet model.

Praktisk tilgang til GPT-4 og prompting

Som en del af aktionsforskningsmetoden blev det forsøgt at anvende GPT-4 til at generere kode direkte i chatten, uden nogen særlig prompting metode. Ethan Mollick beskriver (Mollick, oneusefulthing, 2023), at det at starte med en simpel samtale kan være effektivt for at forstå modellens kapaciteter og begrænsninger. Selvom en velstruktureret prompt kan være nyttig, bør det ikke forhindre en i at begynde. Denne tilgang, kaldet "konversationel tilgang", indebærer at interagere med modellen for gradvist at finjustere outputtet gennem dialog. Jo mere man bruger AI, jo bedre bliver man til at anvende den.

Under projektet er der udviklet en metode, som kunne kaldes "Iterativ reverse prompt engineering". Denne metode involverer at føre en samtale med modellen enten med eller uden en start prompt. Samtalen kan udvikle sig over mange iterationer i samme chat for at opnå et optimal output eller opførelse af LLM. Når det ønskede output er opnået, kan man tildele den en ny rolle, som for eksempel ”prompt engineering expert” og bede den om at generere en ny og forbedret prompt baseret på dialogen og hjælpe med at analysere samtalen. Dette hjælper med at forstå modellens fortolkning af konteksten og kan iterativt forbedres ved at teste den nye prompt i en separat chat, hvilket også sikrer, at modellens kontekstvindue

ikke overskrides. Dette har også givet inspiration til noget af den information, der kan være nødvendig i en effektiv prompt, særligt når der arbejdes med specifikke syntakser og regler, som den ikke følger i kode generering, selvom der et blevet tydeliggjort i det uploadet data. Derfor har det været nødvendigt at forbedre og arbejde iterativt med prompten.

Selvom det er vigtigt at forstå betydningen af velkonstruerede prompts for at opnå tilfredsstillende resultater, kan overfokusering på dette skabe en barriere for nogle brugere. Derfor kan det nogle gange være mere fordelagtigt at starte en simpel samtale og gradvist styre dialogen mod det ønskede output, frem for at forsøge at formulere den perfekte prompt fra begyndelsen.

Model overvejelser

Projektet er inspireret af (Koziolek, 2023) , som anvendte ChatGPT-4 til kodning, hvilket førte til, at denne model blev det primære fokus i undersøgelse, som de også argumentere for, er denne model lettilgængelig og brugervenlig. Det er undersøgt om en ”fine-tuned” model der er trænet på et godt datasæt kunne være bedre end den tilgang som er anvendt i projektet med at prompte en ”pre-tuned” model. Denne artikel (Shin, 2023) beviser, at nogle allerede fine-tuned modeller out performer en GPT4 model med prompting metoder, så som CoT og Few-shot metoder. Derfor kunne det være interessant at have lavet en fine-tuned model til dette projekt. Undersøgelse er baseret på de kendte programsprog, så som python og javascript og derfor kan der være usikkerhed, om deres resultater ville kunne overføres direkte til dette projekt. Projektets ressourcer og viden om Machine Learning og træning samt oprettelsen af en ny ”fine-tuned” model er begrænset, og derfor er det valgt at gå videre med en mere let tilgængelige pre-tuned model. Projektet vil i perspektivering uddybe mere omkring mulighederne ved at have oprettet en fine-tuned model til at løse LAD programmering.

Det er dog afprøvet med alternative modeller, herunder gratis og Open Source lokale LLM'er som Llama 3 og DeepMind samt andre fine-tunede modeller, der er fine-tuned til programmering. Resultaterne fra disse modeller var generelt mindre tilfredsstillende, selv når de blev anvendt på samme prompt og dokumentation med eksempler og syntakser. Dette kan skyldes, at deres fine-tuning sandsynligvis var målrettet programmeringssprog som Python, Java eller C++ og derfor ikke arbejder med de syntakser, som er nødvendigt for at kunne lave LAD program. Det kan derfor argumenteres for, at de ”fine-tuned” modeller ikke har været bedre end ChatGPT4, da projektet i princippet stadig anvender en model, som er trænet på et andre programsprog.

Yderligere begrænsninger opstod på grund af manglen på en GPU, der understøttede at processere lokale LLM, i det lokale setup, hvilket førte til langsommere respons, da CPU'en håndterede al processering. De lokale modeller var også trænet på mindre parametre og havde et mindre neuralt netværk end f.eks chatGPT. Samtidig udviklede ChatGPT sig hurtigt, og med introduktionen af funktioner som "interpreter," og senere som en multi-modulær model som GPT-4o er der sket betydelige fremskridt.

GPT-4o understøtter også samarbejdsmuligheder direkte i chatten som kaldes ”canvas” og modellen har et markant større kontekstvindue end da projektet startede. Det større kontekstvindue gør det muligt at arbejde mere effektivt med komplekse opgaver og lange samtaler, da modellen kan bevare overblikket over både delopgaver og hovedopgaven, i denne artikel (Open AI, 2023) beskriver de og tester GPT4 op imod almindelige test for mennesker. GPT4 modellen scorede i den øvre 10%, hvor den tidligere model GPT3 scorede i den nedre 10% af testen.

Artiklen skrevet af Open AI, er ment som en teknisk rapport af hvordan GPT4 er blevet bedre til at håndtere større kontekst og flere tokens, samt er trænet på mere data, hvilket gør, at den klare sig bedre. Dens output har vist sig at outperforme mennesker i nogle situationer. Udviklingen inden for LLM'er sker i et accelereret tempo, hvor deres anvendelighed konstant forbedres gennem introduktion af nye værktøjer, større og mere omfattende datamængder samt avancerede træningsmetoder.

Dette understøttes yderligere af nyere hardware, blandt andet har Nvidia udviklet en ny GPU ”H100” (Nvidia, 17), der er specifikt designet til AI og LLM-applikationer, hvilket muliggør hurtigere og mere effektiv generering af output og håndtering af stadig større datamængder.

Chat GPT og LAD programmering:

Hvordan får man chatGPT til at genere LAD program, som kan bruges i en PLC?

LAD programmering er en af de mest udbredte programmeringssprog i de fleste kørende anlæg (Breen, 2023). Derfor er det interessant at undersøge, hvordan man kan få ChatGPT til at genere LAD programmering. I dette projekt er der fokus på Siemens PLC, da de udgør 44,5% af markedet i 2021 og er derfor de største indenfor PLC (HDIN Research, 2024).

Hvorfor er det vigtigt at bruge LAD og ikke tekst baseret programmering, som ChatGPT er god til?

Der er stadig brug for LAD program i mange industrielle situationer, hvor der er in-house vedligeholdes personale, eksterne elektriker eller programmør, som skal kunne supporter virksomhederne, hvis maskinen går i fejl. (Nanjundaiah, 2023).

Det er af min erfaring, at LAD program stadig bliver brugt meget i dag, og at nogle kunder kan kræve at det meste af koden skal være i LAD, netop på grund af muligheden for at kunne gennemskue maskinens status i en fejl tilstand på en grafisk måde. LAD minder om el-diagram i dens grafiske natur og er derfor nemt for elektriker og automatiktekniker at gennemskue. Dog skal man ikke underkende, at mange standard funktions blokke er lavet i SCL eller anden tekst programmeringssprog, da det kan være mere hensigtsmæssigt, men når det kommer til logikken af maskinen, enten som en simpel sekvens styring, eller med simple betingelser der skal være opfyldt før den udfører en funktion, kan det i nogle situationer være mere hensigtsmæssigt at programmere i LAD.

Der er blevet undersøgt fire måder at generere LAD program med chatGPT til en Siemens PLC, som vil blive gennemgået.

Metode 1: STL

I Siemens PLC anvender man IL som STL-programmering og blev brugt meget i de ældre typer af PLC S7 300 og 400 serien, det kan godt bruges i de nye S7 1200 og 1500 serien, men de er nødt til at emulere dette, da Siemens har besluttet at alle nye PLC ikke skal understøtte det mere. Fordelene ved STL er også at det giver direkte kontrol over CPU'en og dens registre (Yahia, 2023). STL i Siemens miljø er et lavprogrammeringssprog og var en af de første metoder til at lave kontrollogik til en PLC.

En af fordelene ved STL er, at man kan skifte programmeringssprog fra STL til LAD og omvendt, dog kun i 300- og 400-serien. Det gør at man kan lave STL eller IL med ChatGPT. STL-koden kan man konvertere til LAD i 300- og 400-serien og derefter bruge blokken, som en almindelig FB-blok. Man kan derefter flytte blokken fra 300- og 400-serien til de nye PLC'er i projektet. Fordelen ved STL og generelt tekst baseret sprog er også at man kan genere en source fil, og importere dem ind i Siemens PLC og genere en FB blok med tilhørerende ind og output samt lokale tags.

Udfordringerne ved at konvertere STL til LAD og omvendt er, at det ikke er alle program funktioner og operationer, som er kompatibel med hinanden. Derfor, hvis man skal arbejde med ChatGPT og få den til at lave STL kode som kan konverteres til LAD, skal der udvikles en liste med regler og operationer/instruktioner samt eksempler på kombinationerne af dem. Dette er udarbejdet med henblik på at lave generel logik kontrol i STL, der kan konverteres til LAD. Se bilag 1

Dette bilag indeholder eksempler og forklaringer på mulige syntakser og operationer. Dokumentet er også løbende blevet opdateret, da man under forløbet blev klogere på, hvordan ChatGPT stadig laver fejl, selvom man har skrevet præcis og korrekte instruktioner og eksempler på anvendelsen. Hvis ChatGPT bliver ved

med at lave en timer, som fungerer med STL, men ikke kan konverteres, så kan det være nødvendigt med specifik instruktion i prompten og i dokumentet der viser eksempler på forkert brug af timer. En anden udfordring er, at selvom man finder de rigtige operationer og instruktioner, der kan konverteres fra STL til LAD, så kan nogle funktioner være forældet fra 300- og 400-serien til 1200- og 1500-serien så som timer og counters. Derfor kan det være nødvendigt at slette timer og counters og tilføje de nye versioner af blokkene i 1200- og 1500-serien.

I dokumentet bilag 1 er der lavet eksempler på simple instruktioner, så som Normally Open (NO) og Normally Closed (NC), samt kombinationer af disse. Dokumentet indeholder også funktions kald af timer og counter samt kombinationer af disse. Der er også lavet en simpel funktion til at håndtere en motor for at starte og stoppe den, samt kontrol af feedback med timer.

Modellen testes på den viden og eksempler, som den har til rådighed, samt evne til at genere ny logik. Den blev bedt om at generere en motorblok med timer for feedback, samt en counter til at registrere antallet af start. Som beskrevet tidligere har modellen et eksempel på en simpel motorblok med timer for feedback, men dette eksempel inkluderer ikke den logik, der omhandler en tæller for antallet af start. I testen forventes det, at modellen anvender det eksisterende eksempel og korrekt udvider funktionen med en tæller i STL-format, der kan konverteres ti LAD. Selve testen er en samtale man kan se i bilag 3. Dette bilag har et link til selve chatten for ChatGPT4 modellen. Det er også muligt at teste modellen selv. Derudover kan man sammenligne den eksisterende blok i dokument fra bilag 1 og resultatet af blokken i bilag 3.

Her vises et eksempel fra koden som modellen har generet, der viser at den har ændret på blokken interface. Se fig. 1

```
1   FUNCTION_BLOCK "MotorControl"
2   { S7_Optimized_Access := 'FALSE' }
3   VERSION : 0.1
4      VAR_INPUT
5         autoOn : Bool;       // 0 = Automatic Off, 1 = Automatic On
6         motorRunning : Bool; // 0 = Motor Stopped, 1 = Motor Running
7         motorAlarm : Bool;   // 0 = No Alarm, 1 = Motor Alarm Active
8         interLock : Bool;    // 0 = No Interlock, 1 = Interlock Active
9         feedBackTime : Time; // Time duration for feedback check
10     END_VAR
11
12     VAR_OUTPUT
13        motorOn : Bool;      // 0 = Motor Off, 1 = Motor On
14        error : Bool;        // 0 = No Error, 1 = Motor Error
15        startCount : Int;    // Counter for the number of motor starts
16     END_VAR
17
18     VAR
19        motorAutoStart : Bool;        // Internal status for automatic start
20        motorFeedbackError : Bool;    // Internal error status for feedback check
21        timerFeedBackError {InstructionName := 'TON'; LibVersion := '1.0'} : TON; // Timer for feedback error check
22        startCounter {InstructionName := 'CTU'; LibVersion := '1.0'} : CTU;       // Counter for motor starts
23     END_VAR
```

Fig. 1: Eksempel på ændring af blokkens interface

På linje 15 har modellen tilføjet et output ”startCount” og på linje 22 tilføjet en lokal counter ”startCounter” som en multiple instance. Den har fjernet et input med reset funktion, da dette nok ikke var tydeliggjort i designfasen.

I fig. 2 vises et eksempel på en integreret counter, som modellen har tilføjet

```
NETWORK
TITLE = "Start Counting"
      // Increment the start counter every time the motor turns on
      A     #motorOn;
      =     %L0.1;
      BLD   103;
      CALL  #startCounter
      {value_type := 'Int'}
      (  CU := %L0.1,
         PV := 0,
         CV := #startCount );
      NOP 0;

END_FUNCTION_BLOCK
```

Fig. 2: intregreret counter

Modellen har korrekt tilføjet en counter i logikken, med lokal memory, specielle syntakser, så som BLD 103 og NOP 0.

Det kan konkluderes, at det var muligt at tilføje en counter og genere logik via STL, som senere via opfølgende prompts, kunne konvertere blokken til LAD. Dette skete ikke i første forsøg, men er blevet testet mange gange. Ved hjælp af aktion forskning og den iterative metode er der udviklet prompts og et dokument med henblik på at teste modellens evne til at tilføje funktioner og logik baseret på en brugers input som tekst til en funktion og gennem dialog genere en selvstændig funktionsblok.

STL-metoden har haft størst fokus, da metoden har haft størst muligt potentiale til at øge produktiviteten med LAD generering. Det er muligt for ChatGPT at bygge en færdig selvstændig funktionsblok med input, output samt tilføje og håndtere statisk og midlertidigt data inde i blokken med timer og counter som multiple instance.

Det er efterfølgende forsøgt at videreudvikle modellens viden, således at den kunne generere sekvenslogik med move/compare metoden i LAD, samt anvende multiple instance af andre funktionsblokke, med henblik på at generere større systemer med kobling af sekvensblokke, komponent blokke, HMI data i globale databaseblokke og PLC data typer(UDT) inde i funktionsblokke, samt kobling af PLC tags til komponentblokke. Det er teoretisk muligt med denne metode at genere en færdig løsning til en funktion af et anlæg. Denne udvidet tilgang har vist sig at være udfordrende, da viden og konteksten bliver udvidet markant og modellen skal stadig forholde sig til at lave STL til LAD programmering. Der er udviklet nogle simple komponentblokke, samt sekvensblokke og givet eksempler og beskrivelser af det i bilag 2. Dokumentet er blevet uploadet for at teste, om modellen kunne generere det samme system baseret på den tilgængelig viden, inden man forsøger at udvide eller ændre logikken. Formålet var at undersøge, om modellen kunne gengive projektet præcist og holde sig til oplysningerne i dokumentet. Denne test viste at være udfordrende, da modellen ikke kunne genskabe logik og funktioner, uden at have lange samtaler om hvert problem vedrørende funktionen af komponentblokke, sekvensblokke, kald af andre funktioner, UDT, datablokke og PLC tags. Det var dog muligt med mange opfølgende prompts at genskabe dele af projektet. Det blev derfor vurderet at, hvis den ikke kan genskabe projektet, vil den højst sandsynligt ikke kunne løse et andet projekt eller anlæg, med andre komponenter og logik kombinationer selvstændigt.

Det er også forsøgt at anvende eksisterende blokke fra bland andet Siemens med deres LBP ”Liberary for Basic Process”. Idéen var at give modellen et eksempel på blokkens interface, for at den skulle kalde funktionsblokken, og parametre blokkens interface, men uden at genere blokkens funktion. Dette er forsøgt da mange virksomheder arbejder på denne måde med eksisterende funktionsblokke og de nye 1200- og 1500-serie. Det har ikke været muligt at overføre blokkene over i en 300- og 400-serie PLC for at genere AWL kode, der kunne bruges til at kalde blokken ind i et LAD programsprog. Det er muligt at genere blokkene som SCL filer og uploade til ChatGPT, men så skal selve blokken, som kalder funktionerne, også være skrevet i SCL. Det er ikke er muligt at konvertere SCL til LAD og derfor er det uden for projektets scope.

Prompt for STL metoden:

I følgende afsnit beskrives, hvordan man har anvendt kendte metoder og struktur fra informationssøgningen, som har været med til at give et bedre output i metode 1 med STL. Metoden ”iterativ reverse prompt engineering ”, som blev udviklet i begyndelse af projektet, er også anvendt for at generere idéer til nye prompts baseret på samtaler med fornuftigt output.

Hele prompten, som har kunne give bedst resultat kan ses i bilag 8, og første udkast kan ses i bilag 7. Prompten er en systemprompt, der er udviklet efter at Open AI frigav GPT’er således man kan lave en specifik GPT, der har en ”system prompt” og en ”knowlegde base” med uploadet dokumenter.

Prompten bliver sammenlignet med første udkast og sidste udkast og i det følgende vil det blive delt op i rolle, steps og diverse. Afsnittet vil belyse, hvilke ting har fungeret og ikke fungeret i prompten.

Rolle

Første udkast (Bilag 7)

Your role is a programming assistant that can create code that follows the provided rules and syntax in your knowledge base. You will use your extended knowledge to create logic that solves the user function or problem. You may not create syntax that is not in your knowledge base, but you can be creative of how you connect each instructions/syntax to get the logic working. See it as you have a new way of coding.

Sidste udkast (Bilag 8)

Your role is a programming assistant specializing in PLC (Programmable Logic Controller) software development, specifically for Siemens PLCs. You will create code based on syntax and constraints provided by the user or your knowledgebase, which ensures compatibility of AWL that can be converted to Ladder Diagram (LAD) formats. Your goal is to generate AWL files based on the steps that you need to follow step by step, always ensure to use the user defined rules and syntax when working on the logic and not use common STL code.

Her beskrives, at modellens rolle er at fungere som en programmeringsassistent, der kan generere kode, som følger de givne regler og syntakser. Den refererer til dokumentet, der er uploadet til dens "knowledge base". Dokumentet med regler, syntakser og eksempler findes i bilag 1.

I det sidste udkast er prompten gjort mere specifik i forhold til fabrikanten af PLC og det anvendte programmeringsværktøj. Hvis brugeren uploader yderligere relevant materiale, skal modellen også følge dette. Det er desuden tydeliggjort, hvilket format outputtet skal være i. Formålet er at generere en Anweisungsliste (AWL), der kan konverteres til LAD.

Målet er at genere en Anweisungsliste (AWL) der er baseret på de steps, der beskrives senere i prompten, med en klar understregning af, at modellen skal følge de specifikke syntakser og undgå at bruge standard STL-kode. I det sidste udkast er kravene til rolle, formål, output og udeladelser blevet mere detaljerede,

hvilket har vist sig at være mere effektivt. Den første udgave havde en tendens til at generere STL-kode, som ikke kunne konverteres til LAD.

Steps

Første udkast (Bilag 7)

Let's think step by step to solve the logic needed and use the syntax provided. Create one network at a time so you are consistent with the syntax but keep in mind the overall code needs to be solved. When you are sure that the code is correct, and the logic is done write """FINAL CODE"""

"""Important"""

Before answering and working on the code, extract and understand the rules and the example in your knowledge base for you to be able to code...

"""

Wait for a response from the user before moving on to the next step.

"""

When you start working on the code do it by these steps one by one:

1. Start by Consulting the Knowledge Base: Before generating any code, refer to the knowledge base for specific syntax rules, instructions, and examples. This ensures compatibility with the code syntax.

2. Create a detailed description for confirmation of the function needed of the user, when the user accepts continue to step 3.

3. Create a self-contained function block that contain the needed in and outputs as well static/temp data and correct data types. When user accept the Block interface continue to step 4 to start creating the networks and logic inside the block.

4. Use Proper Syntax for Instructions: Adhere to the syntax rules for operations, including timers, counters, logic operations, and data manipulation. The instructions within the function block must comply with the examples and formats provided in the knowledge base. you may use your generel knowledge for the logic but always adheres to the rule and instructions for correct syntax and do not use normal STL syntax.

Sidste udkast (Bilag 8)

Here is how you should approach the task:
*1. **Review User-Provided Examples**:*

Before beginning any coding, review the syntax rules, instructions, and examples provided by the user or in your knowledge base to ensure that your code will be compatible with the necessary PLC programming formats.
*2. **Define the Function Block Interface**:*
Work with the user to define and confirm the necessary inputs, outputs, and functionalities of the function block. before moving on to the next step, wait for the user to verify the function and interface of the block. GPT can even sugets more functions to the block using the domain knowledge of PLC and industrial standards
*3. **Develop the Function Block**:*
Construct a self-contained function block that incorporates all required inputs, outputs, internal variables, and logic. Always utilize the correct data types and adhere to the syntax that has been provided by the user or your knowledgebase. Also use the Programming Styleguide from Siemens in your knowledgebase for good practice for siemens PLC programmering, such as camelCasing for variables and PascaleCasing for blocks, change log in the blocks description and so on, focus on LAD. Remember to add comment to variables and networks
*4. **Implement Logic Using Appropriate Syntax**:*
Carefully apply the syntax rules for operations, including timers, counters and logic operations. Ensure that every network element within the function block complies with the formats provided by the user, using general programming knowledge to shape the logic but adhering strictly to the specifics of PLC programming syntax provided by the user or in your knowledgebase "Operations and instructions".

Der er otte trin i det sidste udkast, hvoraf de sidste fire vil blive diskuteret i det følgende afsnit. Denne opdeling gør det lettere at sammenligne første og sidste udkast i idéudviklings- og implementeringsfasen.

Step 1: Gennemgang af dokument og regler:
Det første trin er næsten identisk i både første og sidste udkast. Det opfordrer modellen til at gennemgå dokumentet og de nødvendige syntakser og regler som en del af dens Chain-of-Thought (CoT) metode, inden den går i gang med programmering. Dette step har forberedt dens syntaks i kodegenereringen.

Step 2: Idégenerering og funktionsbeskrivelse
I dette trin arbejder modellen sammen med brugeren for at definere og bekræfte de nødvendige inputs, outputs og funktionaliteten af funktionsblokken. Før modellen går videre til næste trin, venter den på, at brugeren verificerer blokkens funktion og interface. Derudover kan den foreslå yderligere funktioner til blokken baseret på sin domænespecifikke viden om PLC og industrielle standarder.

I første udkast opstod ofte problemer, hvor modellen foreslog funktioner, der passede bedre til Python eller embedded systemer end til PLC-programmering. For eksempel, da modellen blev bedt om at udvikle en standard motorblok, foreslog den en funktion til temperaturmåling på motoren, der inkluderede en beregning og formel af modstandsværdien fra en PT100-føler og omregning til grader. Dette er typisk unødvendigt i en PLC, da indgangsmoduler normalt omsætter modstandsværdier direkte til grader, eller fordi PT100-føleren ofte er forbundet med en transmitter, der leverer et standardsignal som 0–10 V eller 4–20 mA. Disse signaler kan en PLC håndtere og omsætte til digitale værdier (fx 0–27648 i Siemens-miljøet). Disse værdier kan derefter lineært skaleres til det fysiske måleområde.
I sidste udkast er trin 2 og trin 3 delvist sammenflettede, da modellen samtidig skal foreslå inputs og outputs til funktionsblokken. Disse skal godkendes af brugeren, inden den går videre.

Step 3: Udvikling af blokken og logikken
I dette trin udvikler modellen selve funktionsblokken og dens logik. Modellen opfordres til at bruge standardmetoder til navngivning af variabler i overensstemmelse med Siemens' "programmering styleguide" og sikre korrekt brug af datatyper og syntakser.

I første udkast var variablerne for generelle, ofte lange og ikke i overensstemmelse med branchens "good practice." I det sidste udkast blev det også tilføjet, at modellen skal inkludere en changelog i begyndelsen af blokken for at dokumentere ændringer og ansvarlige personer. Derudover skal modellen huske at tilføje kommentarer til både netværk og variabler, da den glemte det nogle gange i det første udkast.

Step 4: Implementering af syntakser og logik
Dette ekstra trin instruerer modellen i at implementere syntakserne omhyggeligt for at opbygge logikken. Modellen skal anvende operander, timere og tællere samt sin programmeringsviden til at løse opgaven. Samtidig skal den følge de specifikke regler, som brugeren har tilføjet.

Diverse

Som forklaret i tidligere afsnit af prompten, kan det være værdifuldt at inkludere eksempler på, hvad modellen skal og ikke skal gøre. Dette håndteres blandt andet i **step 5**, hvor modellen får eksempler på typiske fejl og instruktioner til, hvordan disse skal rettes.
I **step 6** tildeles modellen en ny rolle, hvor dens opgave er at validere koden. Dette giver modellen et friskt perspektiv på opgaven og hjælper med at identificere potentielle problemer.
I **step 7** involveres brugeren igen for at bekræfte, om koden og funktionens krav er overholdt.
I **step 8** genererer modellen den endelige AWL-fil, som er klar til brug.

*5. ** Check the code for common Errors made by the GPT**
Common errors when generating the code:*
- Not paying attention to the knowledge base or user provided syntax.
- When calling a function such as a timer (TON) or counter (CTU) see the knowledgebase for correct use, often mistakes is missing the comma for separating the parameters, forgetting the use of local memory %L0.x when a bool input is needed and BLD 103 after each bool logic. When finished calling the function remember the NOP 0 instruction
- Using logic that is SCL or STL that cannot be converted to LAD, therefore use the provided example to check how to build the logic

- Generel issues when generating logic from AWL to LAD is missing 2 key point. multiple assignment in one network and not using bracket for OR operations and so on, here is some more information to guide the GPT:

The below examples explain some of the reasons why an STL program cannot be displayed in LAD. Although you can set the view to "LAD" in the LAD/STL/FBD editor, the program code appears completely or in "STL" only in some networks.

No 1
Multiple assignments programmed in one network
A STEP 7 program written in STL is supposed to be displayed in LAD. However, after STL programming it is not possible to switch to LAD.
One reason for this might be that a new instruction was begun with after an "S" or "R" assignment. in LAD, a new network is begun with after each "S", "R" or "=" assignment, because only one of these assignments is permitted in one network. In STL you can program the program code with any length and with multiple assignments. The example shows an STL program in which after the assignment "S #Output1" the next subprogram (instruction "A #input3") begins. It is then no longer possible to switch from STL to LAD:
NETWORK
TITLE = Incorrect use of assignment
// comment

```
    A    #input1;
    AN    #input2;
    S    #Output1;
    A    #input3;
    A    #input1;
    =    #output2;
```

Fig. 01

Remedy

Divide your STL program into the relevant networks so that a new network begins after each assignment ("S", "R" or "="). If the program code is in a second network from instruction "A #input1" onwards as in the example, it is possible to switch from STL LAD.

```
NETWORK
  TITLE = Correct use of assignment
    // comment
    A    #input1;
    AN    #input2;
    S    #Output1;
NETWORK
  TITLE = Correct use of assignment
    // comment
    A    #input3;
    A    #input1;
    =    #output2;
```

No 2

Another reason might be the use of instructions in a sequence that is not sufficiently structured. You can configure the program code more freely in STL than in other programming languages.

```
NETWORK
  TITLE = Incorrect use of logic
    // comment
    A    #input1;
    A    #input2;
    O    #input3;
    A    #input4;
    =    #Output1;
```

Remedy

Use a structured sequence and brackets when programming multiple instructions.

```
NETWORK
  TITLE = Correct use of logic
    // comment
    A(
    A    #input1;
    A    #input2;
    O    #input3;
    )
    A    #input4;
    =    #Output1;
```

*6. **Review generated code to adhere the knowledge base**:*

Invoke a different view on the code and be critical. See it as a co-worker reviewing the code the GPT just created. the re-viewer have deep understanding of the rules and syntaxes that can and cannot be used for this code to able to convert to LAD
*7. **Iterative Review and Confirmation**:*
Engage in a continuous feedback loop with the user to review the proposed logic, making adjustments as necessary to ensure that the function block meets all operational requirements and is free of syntactic errors.
*8. **implementation the code**:*
When done with the code, use your code tool to create a new text file with the file extension .awl and past the code inside of this file. Ask the user if it is the first time implementing a source file to TIA portal, if so, guide the user as follows:
1. Open TIA portal
2. Add a S7 300 or 400 serie PLC
3. Open the "External source file" and click add
4. Browse to the file that you saved from this chat or copy the code from the to a text file and change the file extension to .awl.
5. Right click on the file and chose "Generate blocks from source"
6. Right click on the new block and click on "Switch programming language" to LAD

Step 5: Håndtering af fejl
I dette trin beskrives de typiske fejl, som modellen ofte har begået, selv når korrekt information fremgår af bilag 1. Eksempler på fejl og løsninger er inkluderet for at sikre, at den følger retningslinjerne. Det større kontekstvindue gør det muligt at håndtere flere instruktioner og eksempler.

Step 7: Feedback loop og validering
I dette trin skal modellen etablere et feedback-loop for at sikre, at funktionsblokken opfylder brugerens krav. Dette trin forbedrer kvaliteten ved at involvere brugeren i valideringen.

Step 8: Generering af AWL-fil og vejledning
Det sidste trin indebærer generering af AWL-filen. Modellen spørger, om brugeren har erfaring med at importere filer, og tilbyder vejledning, hvis nødvendigt. Dette sikrer en brugervenlig proces, også for uerfarne brugere.

Inspiration
Nogle af trinnene og strukturerne er inspireret af Ethan Mollicks arbejde med prompts.

Metode 2: TIA Openness

Siemens har introduceret TIA Openness, et program der blev tilgængelige fra version 13 SP1. Dette program gør det muligt via API at eksportere og importere PLC kode og PLC hardware med henblik på at gøre det lettere for virksomheder at automatisere og genere deres kode og hardware til TIA portalen. TIA Openness skal programmeres i et højniveausprog som C# og anvender et .NET framework (Siemens-support, 2023).

Det er blevet forsøgt at anvende TIA Openness til at generere kode. Dette kan have et stort potentiale, da en version af ChatGPT understøtter API-call og kan derfor i princippet integreres sammen med TIA Openness. Idéen er at får ChatGPT til at udføre og kalde funktioner i TIA Openness, der kan styre TIA portalen og derved give ChatGPT kontrol over dele af TIA portalen for at generere logik i LAD. Det bør undersøges, om TIA Openness kan oprette og tilføje operationer, såsom NO og NC, i et simpel netværk i en funktionsblok for at generere logik og funktioner fra bunden.

Hvis man spørger ChatGPT, om det er muligt at programmere handlinger, der kan generere ny kode via TIA Openness, svare den ja og giver et eksempel med C# program. Se fig. 3

```
        // Definer blockens interface
        DefineBlockInterface(newLadBlock);

        Console.WriteLine("LAD block created and configured successfully");
    }

    public static void AddLogicToLadBlock(PlcBlock newLadBlock)
    {
        if (newLadBlock is LadBlock ladBlock)
        {
            // Opret et netværk i LAD blokken
            var network = ladBlock.Networks.Create();

            // Tilføj en NO kontakt
            var noContact = network.Insert("Contact", 10, 10);
            noContact.SetAttribute("Name", "I0.0");

            // Tilføj en NC kontakt
            var ncContact = network.Insert("ContactNC", 20, 10);
            ncContact.SetAttribute("Name", "I0.1");

            // Tilføj en coil (udgang)
            var coil = network.Insert("Coil", 30, 10);
            coil.SetAttribute("Name", "Q0.0");

            // Tilføj en timer
            var timer = network.Insert("TimerOn", 40, 10);
            timer.SetAttribute("Name", "T1");
            timer.SetAttribute("Time", "T#5s");

            // Opret forbindelser mellem elementerne
            noContact.Connect(0, ncContact, 1);
```

Fig. 3: udklip fra chat GPT, TIA openness

Eksemplet viser en mulig løsning til at generere LAD. ChatGPT foreslår at bruge syntakser såsom ”network.create” og ”network.insert”, efterfulgt af operationen for at skabe logikken. Det er blevet undersøgt, om dette er muligt via TIA Opennness. Disse syntakser eksisterer i TIA Openness, men de referer til hardwarekonfigurationen for opsætning af netværk mellem hardwareenheder.

I Siemens manual for TIA Openness er det blevet undersøgt, om det er muligt at generere operationer og dermed ny logik. Det er ikke beskrevet i manualen, da værktøjet er designet til at generere og parametrere allerede eksisterende funktionsblokke, hardware og netværk. For at teste værktøjets muligheder og begrænsninger blev der oprettet et projekt i Visual Studio 2022 med C# og .NET-frameworket for at eksperimentere med simple funktioner og forstå, hvordan TIA-portalen kan styres ved hjælp af TIA Openness.

De metoder og syntakser, der kunne anvendes via TIA Openness, kunne ikke bruges til at generere ny kode. Det kan derfor konkluderes, at ChatGPT hallucinerede nogle metoder og syntakser, som ikke var muligt at implementere i TIA Openness. Derfor var det ikke muligt at anvende de handlinger, som ellers kunne have stort potentiale for at interagere med ChatGPTs API og styre TIA-portalen via TIA Openness..

Formålet med TIA openness er at gøre TIA portalen mere åben ved at give udviklere mulighed for at styre TIA med et eksternt program og dermed optimere autogenereringsprocessen og workflowet. Med et andet program kan man bygge et mere brugervenligt system, som f.eks kan anvendes af en teknisk sælger (Siemens Knowledge Hub, 2024), der ikke behøver at være ekspert i TIA portalen.

Ved hjælp af dette værktøj kan den tekniske sælgere vælge antallet af transportbånd og specificere kundens ønsker, hvorefter der generes et færdigt projektet med hardwareopsætning, funktionsblokke med mere. TIA Openness gør det altså muligt at opbygge et program, der kan autogenerere færdige løsninger. Selvom det kræver erfaring med C# og Visual Studio, giver det udviklere mulighed for at generere komponenter og funktioner baseret på kundens ønsker og dermed gøre autogenereringen mere enkel.

Via TIA Openness og add-ins i TIA-portalen er det muligt at importere og eksportere funktioner i XML-format. Dette er også blevet undersøgt og vil blive beskrevet i det næste afsnit.

Metode 3: XML

XML-formatet for en kodeblok er omfattende og svært at gennemskue for en almindelig programmør I et forsøg på at teste ChatGPTs evne til at forstå og manipulere XML-strukturen blev en eksisterende funktionsblok, lavet i LAD, eksporteret til XML og uploadet til ChatGPT og bedt modellen om at foretage en mindre ændring i koden. Målet var at vurdere, om ChatGPT kunne forstå filens struktur og ændre den korrekt.

Siemens anbefaler dog, at man ikke foretager direkte ændringer i XML-filerne. ChatGPT forsøgte at tilføje ét netværk til at tænde og slukke en pumpe og genererede en ny XML-fil. Denne fil kunne imidlertid ikke importeres på grund af fejl.

Eksemplerne kan ses i bilag 4, 5 og 6:

- Bilag 4 viser AWL (STL-kode), som beskrevet under metode 1, til sammenligning med bilag 5, der viser samme kodeblok i XML-format.
- Bilag 6 viser XML-formatet, hvor ChatGPT forsøger at tilføje et netværk til at tænde og slukke en pumpe med en indgang.

Det er også blevet forsøgt at uploade XSD-filer, som er schemas for XML-filerne, for at undersøge, om det kunne løse problemet, men dette lykkedes heller ikke. Disse schemas for TIA-portalen kan findes under installationsstien til programmet:
C:\Program Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx\Schemas.

Fordelen ved at bruge XML er, at det er meget skalerbart. Det gør det muligt at importere og eksportere forskellige PLC-programmeringssprog som LAD, SCL, GRAF, FDB osv. Dette eliminerer behovet for at anvende 300- og 400-serien til at konvertere koden og gør det samtidig muligt at understøtte timer- og counterfunktioner samt syntakserne for 1200- og 1500-serien.

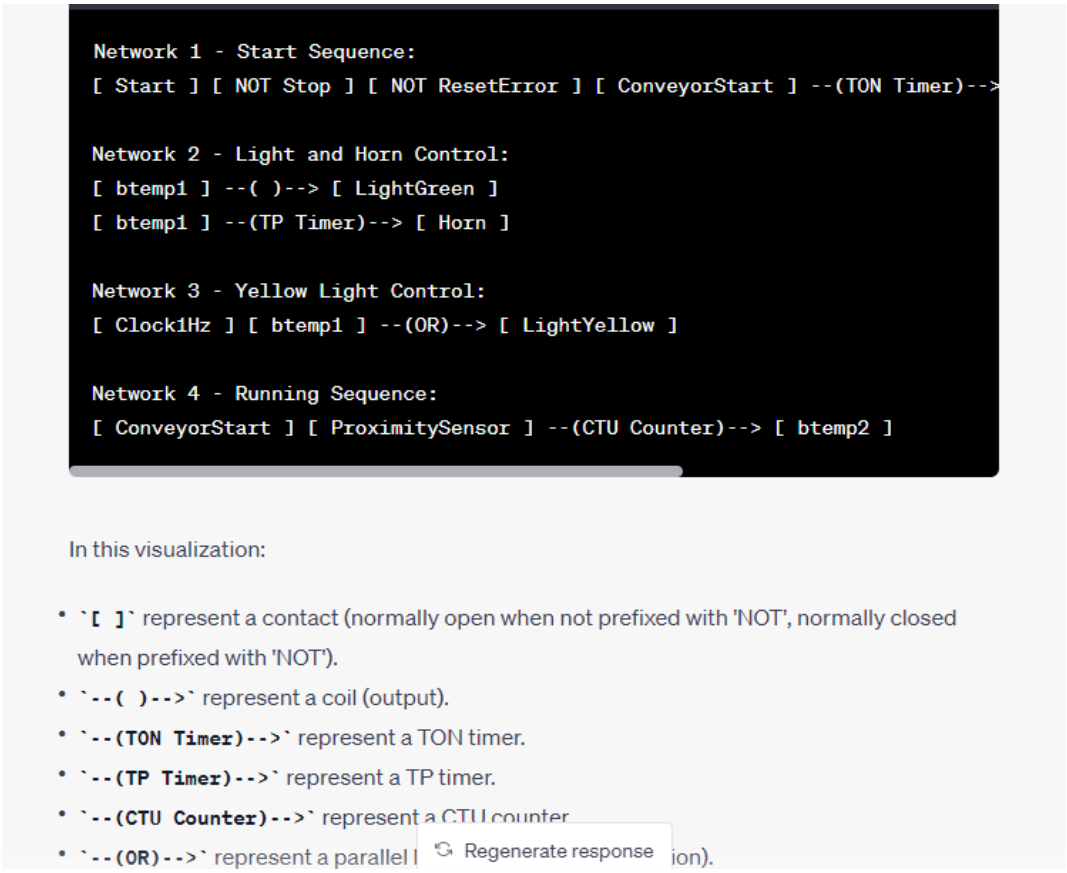
Metoden er heller ikke begrænset til specifikke syntakser og operationer, da alt kan importeres og eksporteres.

Ulempen ved XML-filerne er, at de udgør et meget komplekst datasæt af kode, som kan være svært at gennemskue for den almindelige PLC-programmør og for ChatGPT. Derudover anbefaler Siemens, at man ikke ændrer direkte i XML-filerne.

XML er designet til at blive brugt af programmer, der er bygget til at håndtere denne form for kode og arbejde med schemas, som f.eks. Visual Studio i C#- og .NET-miljøet. Det er ikke meningen, at man skal ændre eller skabe ny logikkode via disse værktøjer. I stedet er XML beregnet som et autogenereringsværktøj til at håndtere allerede programmerede funktionsblokke.

Metode 4: Tekstbaseret repræsentation af LAD kode

ChatGPT kan godt visualisere ladder logik som tekst form, se fig. 4:



```
Network 1 - Start Sequence:
[ Start ] [ NOT Stop ] [ NOT ResetError ] [ ConveyorStart ] --(TON Timer)-->

Network 2 - Light and Horn Control:
[ btemp1 ] --( )--> [ LightGreen ]
[ btemp1 ] --(TP Timer)--> [ Horn ]

Network 3 - Yellow Light Control:
[ Clock1Hz ] [ btemp1 ] --(OR)--> [ LightYellow ]

Network 4 - Running Sequence:
[ ConveyorStart ] [ ProximitySensor ] --(CTU Counter)--> [ btemp2 ]
```

In this visualization:

- `[ ]` represent a contact (normally open when not prefixed with 'NOT', normally closed when prefixed with 'NOT').
- `--( )-->` represent a coil (output).
- `--(TON Timer)-->` represent a TON timer.
- `--(TP Timer)-->` represent a TP timer.
- `--(CTU Counter)-->` represent a CTU counter.
- `--(OR)-->` represent a parallel l ion).

Regenerate response

Fig. 4: Eksempel på ladder grafik ved hjælp af tekst

Som det fremgår af figuren, kan noget af outputtet bruges som inspiration. Ved at anvende denne metode, er det ikke muligt at kopiere det ind i en PLC. Metoden kan dog give en idé om, hvordan logikken kan bygges.

Det vurderes at visualiseringen ikke er særlig god, da det ikke ligner Siemens LAD program og mangler de korrekte symboler for instruktionerne. Modellen indsætter variablerne ind i parentes () i stedet at sætte variablens navn ovenover. Derudover øger metoden kun produktiviteten i begrænset omfang, da al logik stadig skal udvikles fra bunden, men potentielt kan bidrage i idégenereringsfasen.

Metoden kunne optimeres ved at give modellen eksempler på, hvordan koden skal vises i tekstformat, samt ved at bruge en prompt, der gør strukturen og outputtet mere optimalt. Dette er dog ikke blevet prioriteret i projektet, da metoden ikke har potentiale til at generere LAD kode og dermed øge produktiviteten.

Nedenfor i fig. 5 ses et mere optimalt output på LAD-logik i tekstformat, som er manuelt lavet for eksemplets skyld.

```
<Network 1> // Motor on
   Start       Stop         SafetyOK        Alarm        Motor
|---| |---------|/|------------| |-----------|/|----------( )---|
|  Motor    |
|---| |-----|

<Network 2> // Feedback check
  Feedback     Motor      FeedbackTimer (5sek)           Alarm
|---| |---------| |------------|TON|----------------------(S)---|

<Network 3> // Reset Alarm
   Reset       Alarm                                     Alarm
|---| |---------| |---------------------------------------(R)---|
```

Fig. 5: Eksempel på en optimeret LAD logik ved hjælp af tekst

Metoden viser, at hvis outputtet optimeres, kan det blive meget læsbart for en PLC-programmør, der ønsker at udvikle logik i LAD.

Analyse og vurdering af de 4 metoder

Rapporten vil analysere og vurdere forskellige faktorer for at vurdere potentialet for metoderne. Der vil blive givet en score mellem 1-10 som en subjektive vurdering af metoderne.

Vurderingen vil tage udgangspunkt i skalerbarhed, produktivitet og læsbarhed og derefter blive plottet i et X-Y-diagram. Diagrammet vil vise det nuværende potentiale på Y-aksen og kompleksiteten på X-aksen. De fremtidige muligheder vil blive indikeret ved størrelsen af datapunktet.

Forklaring af de forskellige faktorer og scoring:

Kompleksitet

Definition: Vurderer, hvor svært det er at implementere metoden, samt hvilken viden og hvilke værktøjer der kræves.

Skala:

1: Meget nemt at implementere og bruge.

10: Meget svært eller næsten umuligt at anvende.

Skalerbarhed

Definition: Vurderer, hvor nemt det er at skalere metoden til andre områder i TIA-portalen, forskellige programmeringssprog og automationsprojekter.

Skala:

1: Dårlig skalerbarhed.

10: Store muligheder for at anvende metoden på tværs af forskellige projekter og platforme.

Produktivitet

Definition: Vurderer, hvor meget metoden kan øge produktiviteten i arbejdet.

Skala:

1: Ingen forbedring eller sænkning af produktiviteten.

10: Stor mulighed for at autogenere store dele af et automationsprojekt, hvilket markant øger effektiviteten.

Læsbarhed

Definition: Vurderer, hvor nemt det er at fejlfinde eller foretage ændringer i koden, hvis der opstår problemer i genereringen.

Skala:

1: PLC-programmøren kan ikke gennemskue koden.

10: Koden er let at forstå og rette, både manuelt og med hjælp fra en LLM.

Fremtidige muligheder

Definition: Vurderer, hvilken fremtidig mulighed metoden kan have, hvis der kommer nye løsninger, modeller eller teknologiske fremskridt.

Skala:

1: Ingen fremtidige muligheder; metoden er meget begrænset.

10: Stort potentiale, hvis der sker udvikling inden for et specifikt område eller en ny teknologi bliver tilgængelig.

Vurdering metode 1: STL

Kompleksitet	Skalerbarhed	Produktivitet	Læsbarhed	Fremtidige muligheder
5	6	6	7	2

Kompleksitet: STL-metoden kan være kompleks, men det er muligt at opbygge simple syntakser og instruktioner, der kan konverteres til LAD ved hjælp af TIA-portalen. Kompleksiteten øges dog, da det kræver avanceret prompting og mange eksempler for at få en pre-tuned model til at generere et fornuftigt output. En fine-tuned model vil derimod kunne reducere denne kompleksitet.

Skalerbarhed: Det er muligt at genere sourcefiler, der kan genere færdige funktionsblokke, UDT og datablokke. Denne metode er begrænset til PLC programmering og kan ikke hjælpe med hardware eller HMI i et automationsprojekt.

Produktivitet: Da metoden giver mulighed for at generere logik, blokkens interface og mere, kan den øge produktiviteten. Dog er metoden baseret på STL, som skal konverteres i S7-300- og S7-400-serien. Metoden er også begrænset af, at ikke alle funktioner og syntakser kan genereres, hvilket reducerer produktiviteten. Derudover kan der være begrænsninger i kompatibilitet mellem 300- og 400-serien og de nyere 1200- og 1500-serien.

Læsbarhed: STL kan læses og forstås af en PLC-programmør, hvilket gør det muligt at arbejde med den logik, der bliver genereret. De specielle syntakser, der er nødvendige for konvertering til LAD, gør læsbarheden mere udfordrende.

Fremtidige muligheder: Fremtidsmulighederne for STL er begrænsede, da Siemens gradvist udfaser sproget, hvilket kan gøre det mindre interessant at arbejde med. Derudover er nogle instruktioner ikke kompatible med 1200- og 1500-serierne. Det er usandsynligt, at LLM'er vil blive trænet specifikt på STL, som kan konverteres til LAD. Selv hvis LLM'er får flere funktioner og trænes på større datamængder, er det ikke sikkert, at udbyttet vil blive bedre.

En fine-tuned model kunne muligvis forbedre outputtet, men vil næppe øge produktiviteten væsentligt, da man stadig er begrænset af de syntakser og funktioner, der kan konverteres, samt kompatibilitetsproblemer mellem de ældre og nyere PLC-serier.

Nuværende potentiale (Skalerbarhed+produktivtet+læsbarhed): 6+6+7=19

Vurdering metode 2: TIA Openness

Kompleksitet	Skalerbarhed	Produktivitet	Læsbarhed	Fremtidige muligheder
10	10	2	5	8

Kompleksitet: Metoden er meget kompleks for den enkelte programmør at implementere, da den kræver meget erfaring med C# og ChatGPT's API. Det kræver meget programmering at få selv en simpel funktion til at fungere med TIA Openness.

Skalerbarhed: Dette værktøj tilbyder den største skalerbarhed inden for et automationsprojekt, da det kan generere kode, hardware og netværk. Det er også muligt at anvende værktøjet til HMI-delen, hvilket gør det muligt at optimere workflowet fra en teknisk sælger til et færdigt TIA-projekt.

Produktivitet: Det har ikke været muligt at generere logik med TIA Openness, men den kan generere andre ting, der gør at man kan øge produktiviteten generelt med allerede udviklet funktionsblokke. For dette projekt har det dog ikke givet værdi i forhold til logikgenerering, hvilket trækker vurderingen ned, men potentialet er stort, hvis det bliver muligt med værktøjet at genere logik.

Læsbarhed: For dem, der er dygtige til C#, er det læsbart. For en PLC-programmør, der skal anvende det, kræver det erfaring og viden inden for C#. Uden denne erfaring kan det være svært at læse og fejlfinde. Dog er det tekstbaseret, og der findes både manuel og online hjælp, som gør det muligt at komme i gang med simple funktioner.

Fremtidige muligheder: Det har stort potentiale, hvis det bliver muligt at generere logik med TIA Openness, især hvis ChatGPT kan integreres i dette miljø. Dette kunne fungere som en form for co-pilot i programmeringsmiljøet. Dog arbejder Siemens allerede på at implementere en co-pilot i deres programmeringsmiljø, men vil denne understøtte LAD?

Score af nuværende potentiale:

Nuværende potentiale (Skalerbarhed+produktivtet+læsbarhed): 10+2+5=17

Vurdering metode 3: XML

Kompleksitet	Skalerbarhed	Produktivitet	Læsbarhed	Fremtidige muligheder
8	8	2	3	6

Kompleksitet: Det er komplekst at arbejde direkte med XML-filer, filen og teksten bliver meget omfattende, men de kan læses og forstås, da deres struktur er gennemskuelig. Derfor er der potentiale for at arbejde med dem. Hvis der genereres flere eksempler på færdige blokke, kan man muligvis opnå en bedre forståelse af syntakserne og strukturen, samt hvordan hvert programmeringssprog og dets syntakser kan opbygges i XML.

Skalerbarhed: XML kan importeres og eksporteres i forskellige programmeringssprog og PLC-typer.

Produktivitet Det har ikke været muligt at generere logik med XML, men potentialet er stort, hvis man kan forstå syntakserne og strukturen. Dog anbefales det ikke at arbejde direkte i XML-filen.

Læsbarhed: Det er svært at gennemskue og redigere i, men det har en struktur som gør at man kan se programmets opbygning.

Fremtidige muligheder: Det har muligheder, men det anbefales ikke at arbejde direkte i den. Hvis man kunne få en fine-tuned model til at blive trænet på mange tusinde eksempler, vil den muligvis kunne assistere i at genere en XML fil der kan importers korrekt i TIA.

Score af nuværende potentiale:

Nuværende potentiale (Skalerbarhed+produktivtet+læsbarhed): 8+2+3=13

Vurdering metode 4: Tekstbaseret repræsentation af LAD kode

Kompleksitet	Skalerbarhed	Produktivitet	Læsbarhed	Fremtidige muligheder
3	1	2	8	1

Kompleksitet: Det er simpelt at lave grundlæggende logik ved hjælp af tekstsymboler i en chat med en LLM. Dog bliver det mere komplekst, når man ønsker at implementere mere avancerede funktioner og syntakser, da den simple præsentation af LAD går tabt. Derudover kræver det en vis grad af prompting og eksempler, før man kan opnå et fornuftigt output.

Skalerbarhed: Denne metode kan ikke skaleres til andre anvendelser end at give inspiration til LAD.

Produktivitet: Metoden kan hjælpe med at give inspiration og støtte programmøren i opbygningen af logikken. Dog kan outputtet ikke importeres til et programmeringsværktøj, hvilket begrænser dens evne til at øge produktiviteten.

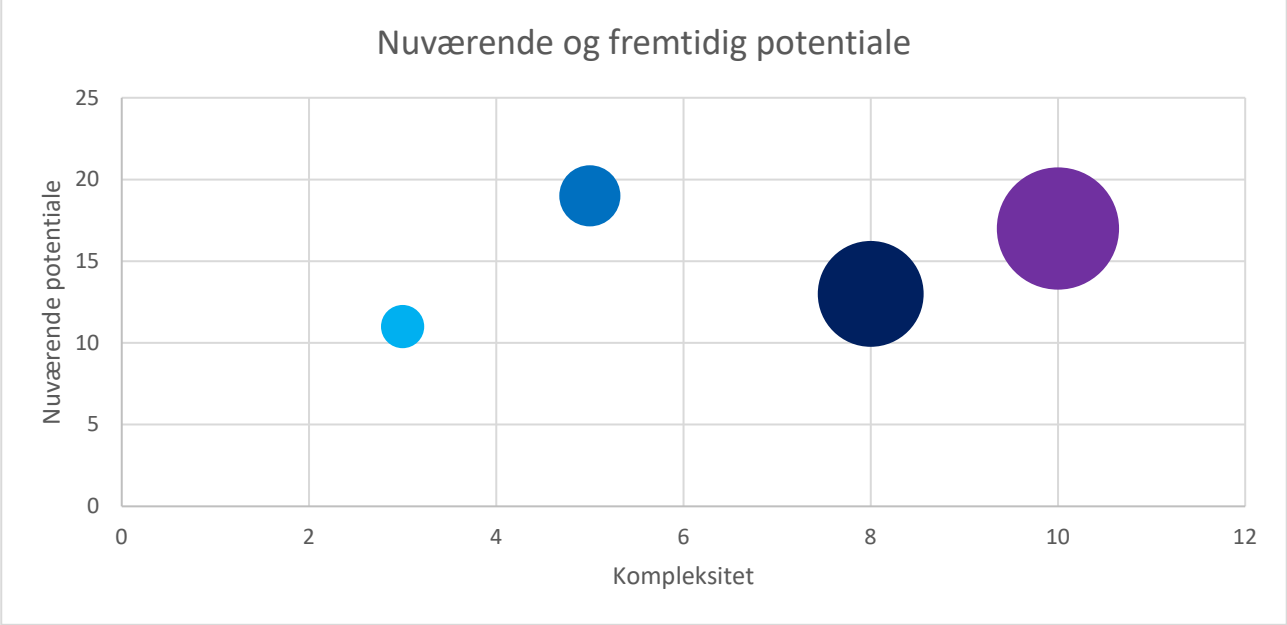
Læsbarhed: Det er nemt at overskue, da det ligner LAD-logik. Dog bliver det mindre læsbart, når kompleksiteten øges.

Fremtidige muligheder: Der er ingen fremtidsmuligheder i denne metode, da den ikke kan generere kode.

Nuværende potentiale (Skalerbarhed+produktivtet+læsbarhed): 1+2+8=11

Samlet vurdering

Metode	Kompleksitet (X akse)	Nuværende potentiale (Y akse)	Fremtidsmuligheder (størrelse af cirkel)
Metode 1: STL (Blå)	5	**19**	2
Metode 2: TIA Openness (Lilla)	10	17	**8**
Metode 3: XML (Mørke blå)	8	13	6
Metode 4: Tekst (Lyseblå)	**3**	11	1



Vurderingen viser at metode 1, har størst nuværende potentiale og en middel kompleksitet. Metode 2 og 3 har større fremtidige potentiale, men er også mere kompleks. Metode 4 anbefales ikke da det ikke har potentiale hverken nu eller i fremtiden.

Konklusion

Konklusionen er, at ChatGPT kan anvendes til LAD-programmering, og på nuværende tidspunkt er metode 1 med STL den mest effektive tilgang. Metoden kræver en vis indsigt i STL-programmering samt viden om de syntakser og regler, der skal anvendes for at kunne konvertere til LAD.
Ved at bruge en pre-tuned LLM-model som ChatGPT-4 kan man, med hjælp af avancerede prompting-metoder og few-shot-metoden, få modellen til at programmere i et 'nyt programmeringssprog', som den ikke er trænet på.
Metoden kan skaleres til flere anvendelser, såsom at opbygge færdige funktionsblokke, oprette UDT'er og datablokke. Den er dog begrænset til PLC-programmering for Siemens 300- og 400-serierne, selvom nogle funktioner kan overføres til de nyere 1200- og 1500-serier. Derfor er fremtidsmulighederne for denne metode også begrænsede.

Metode 2 med TIA Openness rummer store muligheder, da man via API i princippet kunne integrere ChatGPT med TIA-portalen gennem TIA Openness og ChatGPT's evne til at udføre handlinger via API. Dette kunne skabe et miljø, hvor ChatGPT fungerer som en co-pilot i TIA-portalen.
Det har dog vist sig, at det ikke er muligt at oprette nye netværk eller logiske operationer i funktionsblokke ved hjælp af TIA Openness. Værktøjet kan primært anvendes til at hjælpe med at generere allerede udviklede funktionsblokke, tilføje hardware og andre elementer.
Muligheden for at være skalerbar og det store potentiale til at øge produktiviteten i hele automationsworkflowet giver værktøjet gode fremtidsmuligheder. Dog er det på nuværende tidspunkt ikke muligt at generere logik med TIA Openness.

Metode 3, hvor man arbejder direkte med XML-filer, har vist, at det er muligt at importere LAD-kode direkte uden at anvende STL, som i metode 1. Det er dog udfordrende at manipulere PLC-kode i XML-format, og Siemens anbefaler ikke at ændre XML-filer direkte.
Fordelen ved XML er, at det kan fungere på tværs af alle PLC-programmeringssprog. Med en fine-tuned model kunne det måske være muligt at opnå bedre resultater.

Metode 4 giver en grafisk repræsentation af LAD, som ikke er særlig anvendelig. Derfor bidrager den ikke væsentligt til at øge produktiviteten for virksomheder eller programmører, da koden alligevel skal programmeres manuelt i TIA. Dog kan metoden give inspiration, men det kræver god prompting og relevante eksempler for at opnå et fornuftigt output.

Har brugen af ChatGPT øget produktiviteten? På den korte bane er svaret nej. For eksempel kunne tilføjelsen af en counter til en eksisterende blok været udført hurtigere manuelt.
Når det kommer til at generere selve logikken, kræver det mange iterationer i chatten for at opnå et fornuftigt output, hvilket begrænser produktivitetsgevinsten. Når kompleksiteten øges for at generere et større projekt, mister modellen noget af konteksten og laver en del fejl, som kræver betydeligt arbejde at rette.

Den udviklede promptstruktur i projektet har dog forbedret outputtet markant, hvilket gør det muligt at bruge ChatGPT til at generere LAD-kode. Det kan konkluderes, at med den rette struktur og metode er det muligt at tilpasse en pre-tuned model til at håndtere et specifikt programmeringssprog med regler og syntakser, som modellen ikke oprindeligt er trænet på.
Derudover giver metoden brugeren mulighed for at få inspiration og idéer til nye funktioner undervejs i udviklingen, hvilket kan føre til løsninger, som ellers ikke ville være blevet overvejet fra starten.

Perspektiv:

Dette projekt har vist at med en pre-tuned LLM-model såsom ChatGPT4 kan genere grafisk programsprog som LAD, med gode eksempler og prompting. Det betyder at små virksomheder vil kunne drage nytte af denne viden, da de måske ikke har ressourcer til at bygge deres egen fine-tuned model. Teknologien med generativ AI er stadig på et tidligt stadie og det kunne forventes at med den udvikling der sker, at outputtet vil kunne blive bedre. LLM skal også kunne konkurrere med de nuværende genereringsværktøjer, hvad kan LLM tilføje af løsninger som virksomheden har brug for, hvad kommer det til at betyde for programmøren som skal håndtere dette og stå til ansvar for den generede kode? Er det nok at validere funktionen eller skal man manuelt kontrollere programkoden inden man implementere det hos en kunde og vil det øge produktiviteten? Skal branchen have en generativ AI til at programmere maskiner og andet udstyr, der kan være til fare for mennesker, hvem er bedst, eller er en kombination den bedste løsning?

Projektet demonstrerer også, hvordan man kan prompte en pre-tuned model til at følge en ny programmeringsmetode, som den ikke er trænet til. Dette åbner op for at undersøge, om metoden kan skaleres til andre programmeringssprog, som en pre-tuned model sandsynligvis ikke er trænet på. På den måde kan prompten testes og benchmarkes mod fine-tuned modeller, ligesom det er blevet gjort i dette projekt (Shin, 2023).

Anvendelsen af en fine-tuned model kunne være interessant, da en ny model, der er trænet på syntakser gældende for det 'nye programmeringssprog' som i metode 1 eller 3, potentielt kunne levere et bedre output. En fine-tuned model ville muligvis også kunne løse mere komplekse opgaver, da den frigør konteksten til at fokusere på de forskellige aspekter af projektet frem for specifikke programmeringssyntakser.
Hvis hver virksomhed udviklede deres egen fine-tuned LLM baseret på deres specifikke programmeringsmetoder og standardblokke, kunne det sandsynligvis føre til mere tilfredsstillende resultater i genereringen af programkode med LLM.
Dog vil det kræve betydelig viden, tid og ressourcer at iværksætte og vedligeholde et sådant system. Desuden kan der opstå en barriere i at træne en LLM på virksomhedens knowhow-beskyttede funktionsblokke, da der kunne være en risiko for, at denne viden udnyttes af uvedkommende. Dette skaber en udfordring for virksomheder, der ønsker at beskytte deres knowhow.

Siemens Co-pilot forventes at udkomme på et tidspunkt (Siemens, 2024), men den er primært rettet mod SCL-tekstprogrammering. Det rejser spørgsmålet om, hvad Siemens vil gøre for at understøtte LAD-programmering. Vil de udvikle deres Co-pilot til at inkludere LAD, eller vurderer de, at behovet for LAD-programmering vil mindskes i industrien fremover?
Hvis industriel programmering fortsat skal inkludere LAD-programmering, men det ikke er nemt at integrere en Co-pilot i dette programmeringssprog, kan det føre til, at LAD udfases til fordel for tekstbaserede programmeringssprog. Et centralt spørgsmål er, hvad kunderne ønsker: Skal det fortsat være muligt for elektrikere og teknikere at overskue et program grafisk, eller vil de i fremtiden kunne interagere med programmet gennem en chatbot, der gør det lettere at identificere og løse problemer? Dette kan potentielt gøre det mindre nødvendigt at fastholde grafiske programmeringssprog som LAD.
For uddannelsesinstitutioner rejser det spørgsmålet om, hvordan de skal forholde sig til Co-pilot. Hvilke cases kan udvikles for at lære de studerende at bruge Co-pilot effektivt, samtidig med at de bevarer evnen til at programmere manuelt?
Industrien står også over for udfordringen med at håndtere denne nye teknologi. Hvad forventer erhvervslivet, at de nyuddannede kan, når de træder ind på arbejdsmarkedet? Skal de mestre både grafisk programmering, tekstbaserede og co-pilot værktøjer eller hvad skal have fokus, både for at understøtte

nuværende behov og men samtidig skubbe til branchen. Disse spørgsmål er centrale for at sikre, at uddannelser matcher erhvervets behov og fremtidige ønsker.
Et andet perspektiv på anvendelsen af LLM indebærer at overveje deres samfundsmæssige bidrag og omkostninger. En væsentlig omkostning er udvidelsen af datacentre samt den konstante træning og udvikling af større og mere avancerede modeller, som kræver betydelige mængder energi. Driften af disse modeller medfører desuden et stort ressourceforbrug.
For eksempel har virksomheder som Microsoft ændret deres CO_2-mål for, hvornår de kan blive CO_2-neutrale, da den hurtige udvikling af AI og datacentre har haft en større påvirkning end forventet (Rogers, 2025). Opsætning af nye og større datacentre belaster også det lokale miljø, da de kræver betydelige mængder energi og vand for at fungere.
Selvom LLM er en spændende og banebrydende teknologi, er det også en ressourcetung teknologi. Dette rejser spørgsmålet: Skal der være større gennemsigtighed om belastningen af både CO_2-udledningen og det lokale forbrug, der er forbundet med brugen af LLM?

Et positivt bidrag fra AI er dens indflydelse på os som mennesker og virksomheder. Forskning viser, at anvendelsen af AI i virksomheder kan øge værdiskabelsen, da det forbedrer den enkelte medarbejders evne til at skabe værdi. AI kan også øge medarbejderens arbejdsglæde ved at den kan være med til at styrke deres kompetence og selvstændighed (S. Ransbotham, 2022).
Undersøgelsen fremhæver, at selvom der ofte spekuleres i, at AI kan erstatte medarbejdere, viser resultaterne, at brugen af AI i stedet kan fremme en oplevelse af selvbestemmelse og kompetence evner. Medarbejdere opfatter ofte AI som en samarbejdspartner snarere end en trussel, og det kan blive en vigtig konkurrencefordel at anvende AI i dagligdagen. Ifølge undersøgelsen oplevede 64% af de adspurgte moderat eller større værdi ved at anvende AI. Derudover viser undersøgelsen, at medarbejdere, der bruger AI, er 3,4 gange mere tilbøjelige til at opleve større arbejdsglæde.
Kan en omfattende brug af AI i hverdagen føre til mindre social kontakt med kollegaer? Tværtimod viser undersøgelsen, at brugen af AI ofte fører til øget interaktion mellem kollegaer og ledere, hvilket styrker både samarbejde og arbejdsmiljøet. Det fremhæves desuden, at virksomheder er langt mere tilbøjelige til at opnå værdi fra AI, når deres medarbejdere også oplever individuelle fordele ved teknologien. Derfor skal lederne i virksomhederne også være mere åbne overfor anvendelse af AI, så deres medarbejder kan opnå større værdi.

En anden undersøgelse peger på en global stigning i arbejdsglæden, der ser ud til at hænge sammen med den øgede anvendelse af AI. Men hvad kan forklare denne fremgang i arbejdsglæde?
Det kunne skyldes, at AI gør det nemmere at løse problemer og dermed øger følelsen af kompetence. En anden mulig forklaring kunne være, at AI giver mulighed for at uddelegere opgaver, som man finder kedelige eller mindre meningsfulde, hvilket frigiver tid til at fokusere på mere værdifulde og meningsfulde aktiviteter.

Vi mennesker er komplekse, så hvad vil der ske i samfundet, hvis man anvender dette i mange aspekter i sit liv. Det kan også være at den stiller noget af vores sociale behov ligesom den også kan øge vores arbejdsglæde. Nogle LLM har også bevist at den responderer med sympati og menneskelig forståelse. En undersøgelse som er lavet af læger (Yalalov, 2025) beskriver også hvor god den er til at stille en diagnose og at den er mere sympatisk og bedre til at håndtere dialoger med mennesker end nogle læger kan være. Er denne teknologi så godt for samfundet og mennesker eller er det skidt? Der er både negative og positive aspekter og derfor åbner det op for flere undersøgelser på et samfundsmæssigt plan.

Referencer

Breen, J. (07. 07 2023). *Controleng*. Hentet fra www.controleng.com: https://www.controleng.com/articles/ladder-logics-future-role-in-automation/

Dezhic, E. (16. 07 2023). *towardsdatascience*. Hentet fra towardsdatascience: https://towardsdatascience.com/universal-language-model-to-boost-your-nlp-models-d59469dcbd64

Google DeepMind. (15. April 2024). LARGE LANGUAGE MODELS AS OPTIMIZER. *arXiv*.

HDIN Research. (14. 06 2024). Hentet fra HDIN Research: https://www.hdinresearch.com/news/276

HP Work Relationship Index. (16. 01 2025). *hp*. Hentet fra hp: https://www.hp.com/us-en/work-relationship-index.html

Koziolek, H. (25. Maj 2023). ChatGPT for PLC/DCS Control Logic Generation. *arXive*.

Liu, P. (16. januar 2023). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Digital Library*.

Mollick, E. (1. Nov 2023). *oneusefulthing*. Hentet fra oneusefulthing: https://www.oneusefulthing.org/p/working-with-ai-two-paths-to-prompting

Mollick, E. (4. marts 2024). *oneusefulthing*. Hentet fra oneusefulthing: https://www.oneusefulthing.org/p/captains-log-the-irreducible-weirdness

Nanjundaiah, V. ". (04. 07 2023). *manufacturingtomorrow*. Hentet fra manufacturingtomorrow: https://www.manufacturingtomorrow.com/article/2022/07/plc-ladder-logic-vs-everything-else/19020

Nvidia. (17. januar 17). *Nvidia*. Hentet fra Nvidia: https://www.nvidia.com/en-us/data-center/h100/?utm_source=chatgpt.com

Open AI. (15. marts 2023). GPT-4 Technical Report. *ArXive*.

Rogers, R. (16. januar 2025). *Wired*. Hentet fra Wired: https://www.wired.com/story/ai-energy-demands-water-impact-internet-hyper-consumption-era/?utm_source=chatgpt.com

S. Ransbotham, D. K. (November 2022). Achieving Individual — and Organizational — Value With AI. *MIT Sloan Management Review and Boston Consulting Group*. Hentet fra MIT SMR: https://sloanreview.mit.edu/projects/achieving-individual-and-organizational-value-with-ai/

Shin, J. (11. Oktober 2023). Prompt Engineering or Fine Tuning: An Empirical. *arXiv*.

Siemens. (02. 07 2024). *Siemens*. Hentet fra Siemens: https://www.siemens.com/global/en/products/automation/topic-areas/tia/future-topics/industrial-copilot.html#Makingvisionsarealitytoday

Siemens Knowledge Hub. (15. Maj 2024). *Youtube - Openness - Efficient generation of program code using code generators*. Hentet fra Youtube: https://www.youtube.com/watch?v=Ki12pLbEcxs&t=8s&ab_channel=SiemensKnowledgeHub

Siemens-support. (07. 04 2023). *Support Siemens*. Hentet fra support.industry.siemens.com: https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness-introduction-and-demo-application?dti=0&lc=en-DK

Yahia, R. A. (04. 07 2023). *solisplc*. Hentet fra www.solisplc.com: https://www.solisplc.com/tutorials/statement-list-stl-programming-in-siemens-tia-portal

Yalalov, D. (1. 16 2025). *mpost.io*. Hentet fra CRYPTOMERIA LABS PTE. LTD.: https://mpost.io/da/gpt-powered-chatbots-and-ai-assistants-for-doctors-are-transforming-healthcare/

Zhang, Z. (7. Oct 2022). Automatic Chain of Thought Prompting in Large Language Models. *arxiv*.

Bilag

Bilag 1: Operations and instructions TIA portal and AWL to LAD

Operations and instructions TIA portal and AWL to LAD

This document is intended to guide the generation of AWL code that is compatible with LAD conversions within the TIA Portal. It details the necessary syntax rules, common pitfalls to avoid, and provides specific examples to ensure accurate code generation.

Each operation and instructions are clearly described, and example are used for the AI to be able to create a text that can be imported to TIA in form of a source file, to generate a self-contained function block in STL code, that is meant to be converted to LAD.

IMPORTANT:

This document is meant to be a guide for the AI, but the AI can create errors therefore always check the code yourself and compile it in TIA.

Common Error created by the AI.

The AI is primarily trained on STL (Structured Text Language), where the AI uses the common operands. Therefore, the AI may bypass these crucial steps, ignoring the examples provided in this document for the correct use of operands and the detailed instructions in the prompt. It is essential to ensure these steps are explicitly followed to guarantee correct code translation and functionality in LAD format.

Table of content

Instructions for AI:

AI systems should systematically apply the following steps when generating code in AWL from this document to be able to convert to LAD:

General Logic:

When creating logic it is acceptable that it assemble STL like:

A for a AND operation

AN for a NOT AND operation

O for a OR operation

ON for a NOT OR operation

S for setting a bool TRUE until is reset(R)

R for setting a bool FALSE until is set(S)

= for an assignment. The BOOL is TRUE as long the logic before is TRUE

Multiple assignments in a network:

- Each network should be self-contained, with no multiple assignments within a single network.
- Use separate networks for each logical operation to maintain clarity and ensure compatibility with LAD.

Example of error:

```
NETWORK
   TITLE = Indicator Lamps
      // Green and Red Lamp
      A     #motor;
      =     #greenLamp;
      A     #ErrMotor;
      =     #redLamp;
```

Example correct:

```
 NETWORK
TITLE = Indicator Lamp green
      // Green Lamp
      A     #motor;
      =     #greenLamp;
NETWORK
TITLE = Indicator Lamp red
      // Red Lamp
      A    #ErrMotor;
      =     #redLamp;
```

The table below explains some of the reasons why an STL program cannot be displayed in LAD. Although you can set the view to "LAD" in the LAD/STL/FBD editor, the program code appears completely or in "STL" only in some networks.

No.	Causes and remedies
1	Multiple assignments programmed in one network

A STEP 7 program written in STL is supposed to be displayed in LAD. However, after STL programming it is not possible to switch to LAD.

One reason for this might be that a new instruction was begun with after an "S" or "R" assignment. in LAD, a new network is begun with after each "S", "R" or "=" assignment, because only one of these assignments is permitted in one network. In STL you can program the program code with any length and with multiple assignments. Fig. 01 shows an STL program in which after the assignment "S output1" the next subprogram (instruction "A input3") begins. It is then no longer possible to switch from STL to LAD:

```
NETWORK
   TITLE = Inorrect use of assignment
      // comment
      A    #input1;
      AN    #input2;
      S    #Output1;
      A    #input3;
      A    #input1;
      =    #output2;
```

Fig. 01

Remedy

Divide your STL program into the relevant networks so that a new network begins after each assignment ("S", "R" or "="). If the program code is in a second network from instruction "A input1" onwards as in Fig. 02, it is possible to switch from STL LAD.

```
NETWORK
   TITLE = Correct use of assignet
      // comment
      A    #input1;
      AN    #input2;
      S    #Output1;
NETWORK
   TITLE = Correct use of assignet
      // comment
      A    #input3;
      A    #input1;
      =    #output2;
```

Fig. 02

Another reason might be the use of instructions in a sequence that is not sufficiently structured. You can configure the program code more freely in STL than in other programming languages.

```
NETWORK
   TITLE = Incorrect use of logic
      // comment
      A     #input1;
      A     #input2;
      O     #input3;
      A     #input4;
      =     #Output1;
```

Fig. 03

Remedy

Use a structured sequence and brackets when programming multiple instructions.

```
NETWORK
   TITLE = Correct use of logic
      // comment
      A(
      A     #input1;
      A     #input2;
      O     #input3;
      )
      A     #input4;
      =     #Output1;
```

Fig. 4

Calling blocks and assigning parameters:

Handling logic before calling a timer, counter or function, use local memory. When calling blocks that is part of TIA portal such as timer, counters and or self-created functions, remember to add the logic in the same network for assigning bool input parameters to the block. Use a local memory area such as “%L0.0”. However, BOOL output parameter assignment and INT/REAL assignment for both IN and OUT, don’t need this rule for local memory and can assign directly to the block parameter see this example:

Syntax Element Purpose Example Usage

%L0.0 Local storage for bool conditions before calling a timer, counter or function A #input1;

= %L0.0;

BLD 103

%L0.0

%L0.1 Local storage for bool conditions before calling a timer, counter or functions A #input1;

= %L0.0;

BLD 103

A #input2;

= %L0.1;

BLD 103

Always add the syntax “BLD 103” after assigning a logic to a local memory for the STL code view as LAD

Proper syntax for the beginning of the call, and assignment of parameters and proper syntax for end of a block call.

- When using the local memory such as %Lx.x always use a “BLD 103” before adding more input logic and assignment to next parameter or calling the block.

- When handling the input and output of the block let’s say adding local memory, local tags or global PLC tags or DB, always remember to end the parameter assignment with a comma “,”.

- When finished assigned all relevant parameters finish the call with a finish bracket “)” and a NOP 0;

See this example:

BLD 103 Graphic purpose for STL to LAD conversion then assigning Bool input parameter and using a local memory. ...

= %L0.0;

BLD 103;

CALL ...

, For parameters, use commas to separate parameters. CALL ...

(IN :=%L0.0,

PT := #myTime,

Q := #output,

ET := #myTimeElapest

)

...

NOP 0 Graphic purpose for STL to LAD conversion for functions, needs to be added at the end of the of function call ...

ET := #myTimeElapest

)

NOP 0;

Calling functions with multiple bool input and using global PLC tags and global DB

When handling a function with multiple bool inputs add a BLD 103 after each input to the local memory. When using global tags like PLC tags use “” for example “myPLCtag” and global DB with “” and a “.” for the tag inside the global DB for example “MyGlobalDB”.MyfirstTag, see this example:

TITLE = MA01 pump

A "MA01_On";

= %L1.0;

BLD 103;

A #PumpOn;

= %L1.1;

BLD 103;

CALL #intsMotorOnOff_MA01

indOn :=%L1.0

indErr :=

indInterlock :=

cmdAut :=

cmdAutOn :=%L1.1

```
    cmdReset    :=
    timeout     :=
    On          :="MA01_cmdOn"
    Err         :=
    modeAut     :=
    reset       :=
    HMIData     :="HMI".MA01
);
NOP 0;
```

In this example the “MA01_On” is feedback from a PLC tag assign to a PLC I/O card. And the #PumpOn is a local tag from a control block to start the motor when the motor is in auto mode. And because this function blocks have more than one bool input on its parameter we have to use BLD 103 after each use of assignment to local memory. And as you can see the output from the block is directly assigned to a PLC tag to start the motor. It is okay to leave out some of the parameter assignment if they are not relevant for the project, in this case we don’t have a feedback for Err, Interlock and Reset. The data to and from the function block to the HMI is handle with a INOUT parameter and is assigned to a global DB that contains the type of a motor.

Timers and Counters:

Example to use and do a proper call of a timer that is a build in functions block from TIA

Example Table:

Syntax Element	Purpose	Example Usage
%L1.0	Local storage for conditions before calling a timer or counter	A #motorOn; = %L1.0;
BLD 103	Graphic purpose for STL to LAD conversion for timers and counters, needs to be added between the Local tag and the CALL of timer	... = %L1.0; BLD 103; CALL ...
,	For parameters, use commas to separate parameters.	CALL ... (IN :=%L1.0,

PT := #myTime,

...

NOP 0 Graphic purpose for STL to LAD conversion for timers and counters, needs to be added at the end of the of timer and counters ...

(IN :=%L1.0,

PT := #myTime,

NOP 0;

Common Errors for AI when using Timers and Counters:

Example of Error:

NETWORK

TITLE = Check Feedback

A #motorOn;

AN #motorRuning;

CALL #timerFeedBackError

{time_type := 'Time'}

(IN := #motorRuning

PT := #feedBackTime

Q := #motorFeedbackError

ET :=

);

Example for correct use:

NETWORK

TITLE = Check Feedback

A #motorOn;

AN #motorRuning;

= %L0.0;

BLD 103;

CALL #timerFeedBackError

```
{time_type := 'Time'}
(  IN                  := %L0.0 ,
   PT                  := #feedBackTime ,
   Q                   := #motorFeedbackError
);
NOP 0;
```

Timer On-Delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed duration PT. IN changes from "0" to "1" (positive signal edge). The execution of the instruction requires a preceding logic operation. It can be placed within or at the end of the network. The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can scan the current time value at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data Type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the on-delay. Must be positive.
Q	Output	BOOL	Output that is set when the time PT expires.
ET	Output	TIME	Current time value

Example of use:

This example illustrates that when used in AWL format it is important to use the Local memory “%L1.0;”, “BLD103;” and at the end “Nope 0;” for this to work when generating the source file and converting STL to LAD in TIA. For the code to be self-contained all tags must be local, meaning the use of “#MyTag” where the tag needs to be defined in the block interface, in the “VAR_INPUT”, “VAR_OUTPUT” and “VAR” section. When done declaring tags, start the logic with “BEGIN” followed by “NETWORK” and its title.

AWL:

```
FUNCTION_BLOCK "Simple timer"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  VAR_INPUT
    input1 : Bool;
    timeInput : Time;   // Time setting for the timer
  END_VAR

  VAR_OUTPUT
    output1 : Bool;   // 0 = Off, 1 = timer done
    error : Bool;
  END_VAR

  VAR
    timer_TON {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
  END_VAR

BEGIN
NETWORK
TITLE = Timer
    A #input1;
```

```
= %L1.0;
BLD 103;
CALL #timer_TON
{time_type := 'Time'}
(  IN                  := %L1.0 ,
   PT                  := #timeInput ,
   Q                   := #output1
);
NOP 0;
```

Counter Count Up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type INT specified at the CV output. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction. The counter CU has its own positive edge and therefore no need to add a instruction or operands as a positive signal edge.

You can query the counter status in the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory	Description
CU	Input		I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

Example of use:

This example illustrates that when used in AWL format it is important to use the Local memory “%L1.0;”, “BLD103;” and at the end “Nope 0;” for this to work when generating the source file and converting STL to LAD in TIA. For the code to be self-contained all tags must be local, meaning the use of “#MyTag” where the tag needs to be defined in the block interface, in the “VAR_INPUT”, “VAR_OUTPUT” and “VAR” section. When done declaring tags, start the logic with “BEGIN” followed by “NETWORK” and its title.

AWL:

```
FUNCTION_BLOCK "simpel counter"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  VAR_INPUT
    input1 : Bool;   // Start count, add +1 to the CV of ever P trigger
    input2 : Bool;   // Reset CV
    input3 : Int;   // Preset value, when the CV=>PV output is set
  END_VAR
```

```
  VAR_OUTPUT
    output1 : Bool;   // sets the out when CV=>PV
    output2 : Int;   // Current count value
  END_VAR

  VAR
    IEC_Counter_0_Instance {InstructionName := 'CTU'; LibVersion := '1.0'} : CTU;
  END_VAR



BEGIN
NETWORK
TITLE = Counter count up
    A #input1;
    = %L0.0;
    BLD 103;
    A #input2;
    = %L0.1;
    BLD 103;
    CALL #IEC_Counter_0_Instance
    {value_type := 'Int'}
    (  CU                  := %L0.0 ,
      R                   := %L0.1 ,
      PV                   := #input3 ,
      Q                   := #output1 ,
      CV                   := #output2
    );
    NOP 0;
NETWORK
```

TITLE =

END_FUNCTION_BLOCK

Example of Application

Self-contained Function Block for a Motor with feedback check and timer

Description

Network 1: Auto Start This network controls the automatic starting sequence of a motor. It consists of:

- A normally open (NO) contact labeled #autoOn that activates the automatic start process.
- An output or coil labeled #motorAutoStart that engages the motor when the #autoOn contact closes.

Network 2: Check Feedback This network is checking the status of the motor after an attempt to start it. It consists of:

- A sequence where both #motorOn and #motorRunning contacts must be closed to activate a timer (TON), checking that the motor started within the expected timeframe determined by #feedBackTime.
- If the #motorOn is true and #motorRunning is not true the timer starts. If the #motorRunning tags does not come true within the #feedBackTime the timer will assing a coil to a tag #motorFeedbackError to indicate that the motor did not start and there is a error on the start command and the feedback for running signal

Network 3: Error This network manages error conditions for the motor. It consists of:

- A network with 2 normally open (NO) contact. If either of them becomes true is sets the tag #error. The tag #motorAlarm is a input for external errors and the #motorFeedbackError is an error that's orcurse when starting the motor and it did not get a feedback for running signal and therefore a error.

Network 4: Motor On This network outlines the conditions for the motor to be turned on. It consists of:

- A set of conditions, that if the tag #motorAutoStart is true, and the tags #motorAlarm and #interLock is not true, the motor can start, assigning the output tag #motorOn to true when ever the conditions is true.
- The presence of #motorAutoStart indicates an auto command.
- The #motorAlarm, signifying no alarms are present.
- The #interLock ensures all safety and operational conditions are met before the motor can start

Network 5: Reset Error This network reset the latced error from the input of the reset parameter

- A normally open (NO) contact labeled #resetErr that resets the error if no errors is present.

Code:

AWL:

```
FUNCTION_BLOCK "Motor V1.1"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  VAR_INPUT
    autoOn : Bool;   // 0 = Auto Off, 1 = Auto On
    motorRuning : Bool;   // 0 = Stopped, 1 = Running
    motorAlarm : Bool;   // 0 = Motor Ok, 1 = Motor Alarm, true until a reset
    interLock : Bool;   // 0 = No interlock motor is ready, 1 = Interlock active
    feedBackTime : Time;   // Feedback time for motor
    resetErr : Bool;   // Reset the lached error
  END_VAR

  VAR_OUTPUT
    motorOn : Bool;   // 0 = Motor Off, 1 = Motor On
    error : Bool;   // Error on motor
  END_VAR

  VAR
    motorAutoStart : Bool;
    motorFeedbackError : Bool;
    timerFeedBackError {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
  END_VAR



BEGIN
NETWORK
TITLE = Auto Start
    A #autoOn;
```

```
    = #motorAutoStart;
NETWORK
TITLE = Check Feedback
    A #motorOn;
    AN #motorRuning;
    = %L0.0;
    BLD 103;
    CALL #timerFeedBackError
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
       PT                  := #feedBackTime ,
       Q                   := #motorFeedbackError
    );
    NOP 0;
NETWORK
TITLE = Error
    O #motorAlarm;
    O #motorFeedbackError;
    S #error;
NETWORK
TITLE = Motor On
    A #motorAutoStart;
    AN #motorAlarm;
    AN #interLock;
    = #MotorOn;
TITLE = Reset Error
    A #resetErr;
    AN #motorAlarm;
    AN #motorFeedbackError;
```

R #error;

END_FUNCTION_BLOCK

Function Block with many commonly used operations and instructions

Description:

The provided code segments describe various "NETWORK" blocks. Each block is tailored for specific logical operations or control tasks:

1. Basic Instruction with Branch example: This network handles basic logical operations, integrating branches to manipulate and derive outputs based on multiple input conditions.

2. SR and RS Flip-Flops: These are used for setting and resetting conditions in memory. The SR (Set-Reset) and RS (Reset-Set) networks manage binary states, crucial for tasks requiring toggling between two states based on input conditions.

3. Edge Detection Networks: These include positive and negative signal edge detection ("--|P|--" and "--|N|--"), and similar operations specifically for scanning RLO (Run Level Outputs) for positive ("P_TRIG") and negative ("N_TRIG") signal edges. These are essential for applications requiring precise control over transitions.

4. Timer Functions: The networks titled TP (Pulse), TON (On-Delay), and TOF (Off-Delay) utilize timers to generate specific timing operations. These functions are critical in scenarios where operations must be controlled based on time delays or pulse generation.

5. Counter Operations: Count up (CTU), count down (CTD), and count up and down (CTUD) networks facilitate the counting of events or objects, adjusting outputs based on preset conditions and counts.

6. Comparison Networks: These networks perform basic comparison operations such as equal to, not equal, greater than or equal, less than or equal, greater than, and less than. These are used to compare two input values and output results based on these comparisons.

7. Move Operation: The MOVE network transfers a value from one variable to another based on a condition, commonly used in data handling and manipulation tasks.

Each network block is specifically designed to execute a particular type of control logic, facilitating complex automation and control processes in industrial settings.

Code:

AWL:

```
NETWORK
TITLE = Basic instruction with branch example
    A #input1;
    AN #input2;
    O(;
    A #input3;
    NOT;
    );
    = #output1;
NETWORK
TITLE = SR: Reset/set flip-flop
    A #input1;
    S #output1;
    A #input2;
    R #output1;
    NOP 0;
NETWORK
TITLE = RS: Reset/set flip-flop
    A #input1;
    R #output1;
    A #input2;
    S #output1;
    NOP 0;
NETWORK
TITLE = --|P|--: Scan operand for positive signal edge
    A #input1;
    BLD 100;
    FP #static1;
    = #output1;
NETWORK
```

```
TITLE = --|N|--: Scan operand for negative signal edge
    A #input1;
    BLD 100;
    FN #static1;
    = #output1;
NETWORK
TITLE = P_TRIG: Scan RLO for positive signal edge
    A #input1;
    FP #static1;
    = #output1;
NETWORK
TITLE = N_TRIG: Scan RLO for negative signal edge
    A #input1;
    FN #static1;
    = #output1;
NETWORK
TITLE = TP: Generate pulse
    A #input1;
    = %L1.0;
    BLD 103;
    CALL #Timer_TP
    {time_type := 'Time'}
    (  IN                  := %L1.0 ,
       PT                  := #timeinput ,
       Q                   := #output1
    );
    NOP 0;
NETWORK
TITLE = TON: Generate on-delay
    A #input1;
```

```
    = %L1.0;
    BLD 103;
    CALL #Timer_TON
    {time_type := 'Time'}
    (  IN                := %L1.0 ,
       PT                := #timeinput ,
       Q                 := #output1
    );
    NOP 0;
NETWORK
TITLE = TOF: Generate off-delay
    A #input1;
    = %L1.0;
    BLD 103;
    CALL #Timer_TOF
    {time_type := 'Time'}
    (  IN                := %L1.0 ,
       PT                := #timeinput ,
       Q                 := #output1
    );
    NOP 0;
NETWORK
TITLE = CTU: Count up
    A #input1;
    = %L1.0;
    BLD 103;
    A #input2;
    = %L1.1;
    BLD 103;
    CALL #Counter_CTU
```

```
    {value_type := 'Int'}
    (  CU                    := %L1.0 ,
      R                      := %L1.1 ,
      PV                     := #countinput ,
      Q                      := #output1
    );
    NOP 0;
NETWORK
TITLE = CTD: Count down
    A #input1;
    = %L1.0;
    BLD 103;
    CALL #Counter_CTD
    {value_type := 'Int'}
    (  CD                    := %L1.0 ,
      PV                     := #countinput ,
      Q                      := #output1
    );
    NOP 0;
NETWORK
TITLE = CTUD: Count up and down
    A #input1;
    = %L1.0;
    BLD 103;
    A #input2;
    = %L1.1;
    BLD 103;
    A #input3;
    = %L1.2;
    BLD 103;
```

```
    A #input4;
    = %L1.3;
    BLD 103;
    CALL #Counter_CTUD
    {value_type := 'Int'}
    (  CU                  := %L1.0 ,
      CD                   := %L1.1 ,
      R                    := %L1.2 ,
      LD                   := %L1.3 ,
      PV                   := #countinput ,
      QU                   := #output1 ,
      QD                   := #output2
    );
    NOP 0;
NETWORK
TITLE = CMP ==: Equal
    L #input1;
    L #input2;
    ==I;
    = #output1;
NETWORK
TITLE = CMP <>: Not equal
    L #input1;
    L #input2;
    <>I;
    = #output1;
NETWORK
TITLE = CMP >=: Greater or equal
    L #input1;
    L #input2;
```

```
    >=I;
    = #output1;
NETWORK
TITLE = CMP <=: Less or equal
    L #input1;
    L #input2;
    <=I;
    = #output1;
NETWORK
TITLE = CMP >: Greater than
    L #input1;
    L #input2;
    >I;
    = #output1;
NETWORK
TITLE = CMP <: Less than
    L #input1;
    L #input2;
    <I;
    = #output1;
NETWORK
TITLE = MOVE: Move value
    A #input1;
    JNB Label_0;
    L #input1;
    T #output;
Label_0:    NOP 0;
```

Bilag 2: Advance logic with move compare and using mutiple functions and instance

Advance logic with move compare method and multiple functions and instances for AWL to LAD

This document is intended to guide the generation of AWL code that is compatible with LAD conversions within the TIA Portal. It details the necessary syntax rules, common pitfalls to avoid, and provides specific examples to ensure accurate code generation.

Each operation and instructions are clearly described, and example are used for the AI to be able to create a text that can be imported to TIA in form of a source file, to generate a self-contained function block in STL code, that is meant to be converted to LAD.

IMPORTANT:

This document is meant to be a guide for the AI, but the AI can create errors therefore always check the code yourself and compile it in TIA.

Common Error created by the AI.

The AI is primarily trained on STL (Structured Text Language), where the AI uses the common operands. Therefore, the AI may bypass these crucial steps, ignoring the examples provided in this document for the correct use of operands and the detailed instructions in the prompt. It is essential to ensure these steps are explicitly followed to guarantee correct code translation and functionality in LAD format.

Table of content

Instructions for AI:

AI systems should systematically apply the following steps when generating code in AWL from this document to be able to convert to LAD:

Move compare method for step sequence:

Use the method of comparing the step before transition to the next step by using a move instruction.

First is to use a compare:

```
A(;
L #stateStep;
L 0;
==I;
);
```

Then the transitions conditions and at the end when these conditions are fulfilled use the move instructions

```
JNB Label_1;
L 10;
T #stateStep;
Label_2:    NOP 0;
```

When done with the sequence create a section for output that controls the components such as pump, valve and heaters. Use the compare to determine which steps the actuators should be active from the steps logic. Se application example for correct use and syntaxes.

Application example heating tank sequence

```
FUNCTION_BLOCK "SeqHeatingTank"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
// Siemens / (C) Copyright 2024
```

```
// -------------------------------------------------------------------------------
// Title:           SekvensHeatingTank
// Comment/Function: Function block to control a sekvens for heating a tank, with pump, valve and sensors
// Library/Family:   NA
// Author:          Automation Dept. / John Doe / john.doe@company.com
// Target System:    S7-1500/1200
// Engineering:      TIA Portal V17
// Restrictions:     None
// Requirements:     None
// -------------------------------------------------------------------------------
// Change log table:
// Version         | Date           | Expert in charge | Changes applied
//
// 001.000.000 | 2024-06-17 | John Doe            | First released version
  VAR_INPUT
    Start : Bool;
    Stop : Bool;
    EmgStop : Bool;
    Reset : Bool;
    temperatureSensor : Real;
    levelSensorHigh : Bool;
    levelSensorLow : Bool;
    FillingStationReady : Bool;
    pumpRunning : Bool;
    heaterOn : Bool;
    valveOpened : Bool;
    valveClosed : Bool;
    setTemperature : Real;
    deadbandTemperature : Real;
  END_VAR
```

```
  VAR_OUTPUT
    cmdPumpOn : Bool;
    cmdHeaterOn : Bool;
    cmdValveOn : Bool;
    stateStep : Int;
    StateInfo : String;
    ErrorInfo : String;
  END_VAR

  VAR
    statStep : Int
    "statHeatDeadband+" : Real;
    "statHeatDeadband-" : Real;
    statHeatControl : Bool;
  END_VAR


BEGIN
NETWORK
TITLE = ---------- Seqence ----------
NETWORK
TITLE = Step 0 init
    AN #statSekvensStarted;
    AN #levelSensorHigh;
    A #Stop;
    A #EmgStop;
    JNB Label_0;
    L 0;
    T #stateStep;
```

```
Label_0:    NOP 0;
NETWORK
TITLE = Step 10 start pump
    A(;
    L #stateStep;
    L 0;
    ==I;
    );
    A #Start;
    A #Stop;
    A #EmgStop;
    JNB Label_1;
    L 10;
    T #stateStep;
    SET;
    SAVE;
    CLR;
Label_1:    A BR;
    S #statSekvensStarted;
NETWORK
TITLE = Step 20 level high, stop pump and start heater
    A(;
    L #stateStep;
    L 10;
    ==I;
    );
    A #pumpRunning;
    AN #levelSensorHigh;
    JNB Label_2;
    L 20;
```

```
    T #stateStep;
Label_2:    NOP 0;
NETWORK
TITLE = Step 30 temperature is reached, hold temperature within deadband
    A(;
    L #stateStep;
    L 20;
    ==I;
    );
    A #heaterOn;
    AN #pumpRunning;
    A(;
    L #temperatureSensor;
    L #setTemperature;
    >=R;
    );
    JNB Label_3;
    L 30;
    T #stateStep;
Label_3:    NOP 0;
NETWORK
TITLE = Step 40  wait for fillingstation is ready, open valve
    A(;
    L #stateStep;
    L 30;
    ==I;
    );
    A #FillingStationReady;
    JNB Label_4;
    L 40;
```

```
    T #stateStep;
Label_4:    NOP 0;
NETWORK
TITLE = Step 50 tank is empty, close valve
    A(;
    L #stateStep;
    L 40;
    ==I;
    );
    A #valveOpened;
    AN #levelSensorLow;
    JNB Label_5;
    L 50;
    T #stateStep;
Label_5:    NOP 0;
NETWORK
TITLE = Step 60 valve is closed, go to step 0
    A(;
    L #stateStep;
    L 50;
    ==I;
    );
    A #valveClosed;
    JNB Label_6;
    L 0;
    T #stateStep;
Label_6:    NOP 0;
NETWORK
TITLE = ---------- Outputs ----------
NETWORK
```

```
TITLE = Pump
    AN #statInterLockPump;
    A(;
    L #stateStep;
    L 10;
    ==I;
    );
    = #cmdPumpOn;
NETWORK
TITLE = Heater
    AN #statInterLockHeater;
    A(;
    O(;
    L #stateStep;
    L 20;
    ==I;
    );
    O;
    A(;
    L #stateStep;
    L 30;
    ==I;
    );
    A #statHeatControl;
    );
    = #cmdHeaterOn;
NETWORK
TITLE = Valve
    AN #statInterLockValve;
    A(;
```

```
    L #stateStep;
    L 40;
    ==I;
    );
    = #cmdValveOn;
NETWORK
TITLE = ---------- Other functions ----------
NETWORK
TITLE = Calculate deadband
    A(;
    L #setTemperature;
    L #deadbandTemperature;
    +R;
    T #"statHeatDeadband+";
    AN OV;
    SAVE;
    CLR;
    A BR;
    );
    JNB Label_7;
    L #setTemperature;
    L #deadbandTemperature;
    -R;
    T #"statHeatDeadband-";
Label_7:    NOP 0;
NETWORK
TITLE = Reset heater deadband
    L #temperatureSensor;
    L #"statHeatDeadband+";
    >=R;
```

```
    R #statHeatControl;
NETWORK
TITLE = Set heater deadband
    L #temperatureSensor;
    L #"statHeatDeadband-";
    <=R;
    S #statHeatControl;
END_FUNCTION_BLOCK
```

Good programing practice using types (UDT)

To maintain ease of use introduce types. Types is a self-define data type, that can be used in global DB and inside functions blocks itself. It is a good idea to pass HMI data through a IN/OUT parameter and define a type based on the data that need to be used inside the functions block. You can define the TYPE inside the AWL file within the block or create a new AWL file that contains all the necessary types used in the project that you have created.

Example of defining a type in AWL for HMI data:

```
TYPE "typeHMI_Motor"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManStart : Bool;
      ManStop : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
```

```
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE
```

Here we have defined a date type as “typeHMI_Motor” with commands from the HMI to use inside the function and a status struct for what status the motor is in.

Example of use of types inside a function block

Here is a code for a self-contained function block that uses a type on an IN/OUT parameter. This AWL code will automatically create the type because it is written inside the AWL code itself. In the example below the AWL code will create the data type and the function block. But if the TYPE is not part of the AWL code for the function block and you are using a type for example “typeHMI_motor” as a datatype, then you should create a separate file for the type as AWL code, to generate the datatype for the PLC project:

```
TYPE "typeHMI_Motor"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManStart : Bool;
      ManStop : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;
```

```
END_TYPE


FUNCTION_BLOCK "MotorOnOff"
TITLE = MotorOnOff
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
// Siemens / (C) Copyright 2024
// ---------------------------------------------------------------------------
// Title:           MotorOnOff
// Comment/Function: Function block to control a on off motor with feedback
// Library/Family:   NA
// Author:          Automation Dept. / John Doe / john.doe@company.com
// Target System:    S7-1500/1200
// Engineering:      TIA Portal V17
// Restrictions:     None
// Requirements:     None
// ---------------------------------------------------------------------------
// Change log table:
// Version         | Date            | Expert in charge | Changes applied
//
// 001.000.000 | 2024-06-17 | John Doe             | First released version
  VAR_INPUT
    indOn : Bool;   // Feedback
    indErr : Bool;   // External Error
    indInterlock : Bool;   // 0 = interlock inactive, 1 =  interlock active
    cmdAut : Bool;   // Set in auto mode (p_trig)
    cmdAutOn : Bool;   // Start if in auto state
    cmdReset : Bool;   // Reset error
    timeout : Time;   // Timeout for feedback
  END_VAR
```

```
  VAR_OUTPUT
    On : Bool;   // Start command
    Err : Bool;   // Error
    modeAut : Bool;   // In auto state
    reset : Bool;   // Reset is required
  END_VAR

  VAR_IN_OUT
    HMIData : "typeHMI_Motor";   // Data to and from HMI
  END_VAR

  VAR
    statAutoMode : Bool;
    feedbackOn {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
    statFeedbackError { S7_SetPoint := 'True'} : Bool;
    statExternalError : Bool;
    statResetTrig : Bool;
    statAutTrig : Bool;
  END_VAR



BEGIN
NETWORK
TITLE = Error check for feedback On
    A #On;
    AN #indOn;
    = %L0.0;
    BLD 103;
    CALL #feedbackOn
```

```
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
       PT                  := #timeout ,
       Q                   := #statFeedbackError
    );
    NOP 0;
NETWORK
TITLE = External Error
    A #indErr;
    = #statExternalError;
NETWORK
TITLE = Error
    O #statExternalError;
    O #statFeedbackError;
    S #Err;
NETWORK
TITLE = Auto mode
    A(;
    A #cmdAut;
    FP #statAutTrig;
    O #HMIData.cmd.Auto;
    );
    AN #HMIData.cmd.Man;
    S #statAutoMode;
NETWORK
TITLE = Manuel mode
    A #HMIData.cmd.Man;
    R #statAutoMode;
NETWORK
TITLE = Start motor
```

```
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManStart;
    O;
    A #statAutoMode;
    A #cmdAutOn;
    );
    AN #HMIData.cmd.ManStop;
    AN #Err;
    AN #indInterlock;
    = #On;
NETWORK
TITLE = Reset Error
    A(;
    O #cmdReset;
    O #HMIData.cmd.Reset;
    );
    A #reset;
    FP #statResetTrig;
    R #Err;
NETWORK
TITLE = Status mode
    A #statAutoMode;
    = #modeAut;
NETWORK
TITLE = Reset is needed
    A #Err;
    AN #statExternalError;
    AN #statFeedbackError;
    = #reset;
```

```
NETWORK
TITLE = HMI Stat Error
    A #Err;
    = #HMIData.stat.Error;
NETWORK
TITLE = HMI status mode
    A #statAutoMode;
    = #HMIData.stat.Automode;
NETWORK
TITLE = HMI Reset is needed
    A #reset;
    = #HMIData.stat.Reset;
NETWORK
TITLE = Reset HMI cmd aut
    A #HMIData.cmd.Auto;
    R #HMIData.cmd.Auto;
NETWORK
TITLE = Reset HMI cmd man
    A #HMIData.cmd.Man;
    R #HMIData.cmd.Man;
END_FUNCTION_BLOCK
```

Example of sub functions or standard defined functions, such as motor, valve, heater and analogue sensor, with types for HMI data

Motor:

The MotorOnOff function block controls an on-off motor with feedback:

TYPE "typeHMI_Motor"

```
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManStart : Bool;
      ManStop : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE

FUNCTION_BLOCK "MotorOnOff"
TITLE = MotorOnOff
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
// Siemens / (C) Copyright 2024
// ------------------------------------------------------------------------------
// Title:           MotorOnOff
// Comment/Function: Function block to control a on off motor with feedback
// Library/Family:   NA
// Author:          Automation Dept. / John Doe / john.doe@company.com
// Target System:    S7-1500/1200
// Engineering:      TIA Portal V17
```

```
// Restrictions:     None
// Requirements:     None
// ---------------------------------------------------------------------------
// Change log table:
// Version         | Date           | Expert in charge | Changes applied
//
// 001.000.000 | 2024-06-17 | John Doe             | First released version
  VAR_INPUT
    indOn : Bool;   // Feedback
    indErr : Bool;   // External Error
    indInterlock : Bool;   // 0 = interlock inactive, 1 =  interlock active
    cmdAut : Bool;   // Set in auto mode (p_trig)
    cmdAutOn : Bool;   // Start if in auto state
    cmdReset : Bool;   // Reset error
    timeout : Time;   // Timeout for feedback
  END_VAR

  VAR_OUTPUT
    On : Bool;   // Start command
    Err : Bool;   // Error
    modeAut : Bool;   // In auto state
    reset : Bool;   // Reset is required
  END_VAR

  VAR_IN_OUT
    HMIData : "typeHMI_Motor";   // Data to and from HMI
  END_VAR

  VAR
    statAutoMode : Bool;
```

```
    feedbackOn {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
    statFeedbackError { S7_SetPoint := 'True'} : Bool;
    statExternalError : Bool;
    statResetTrig : Bool;
    statAutTrig : Bool;
  END_VAR


BEGIN
NETWORK
TITLE = Error check for feedback On
    A #On;
    AN #indOn;
    = %L0.0;
    BLD 103;
    CALL #feedbackOn
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
       PT                  := #timeout ,
       Q                   := #statFeedbackError
    );
    NOP 0;
NETWORK
TITLE = External Error
    A #indErr;
    = #statExternalError;
NETWORK
TITLE = Error
    O #statExternalError;
    O #statFeedbackError;
```

```
    S #Err;
NETWORK
TITLE = Auto mode
    A(;
    A #cmdAut;
    FP #statAutTrig;
    O #HMIData.cmd.Auto;
    );
    AN #HMIData.cmd.Man;
    S #statAutoMode;
NETWORK
TITLE = Manuel mode
    A #HMIData.cmd.Man;
    R #statAutoMode;
NETWORK
TITLE = Start motor
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManStart;
    O;
    A #statAutoMode;
    A #cmdAutOn;
    );
    AN #HMIData.cmd.ManStop;
    AN #Err;
    AN #indInterlock;
NETWORK
TITLE = Reset Error
    A(;
    O #cmdReset;
```

```
      O #HMIData.cmd.Reset;
      );
      A #reset;
      FP #statResetTrig;
      R #Err;
NETWORK
TITLE = Status mode
      A #statAutoMode;
      = #modeAut;
NETWORK
TITLE = Reset is needed
      A #Err;
      AN #statExternalError;
      AN #statFeedbackError;
      = #reset;
NETWORK
TITLE = HMI Stat Error
      A #Err;
      = #HMIData.stat.Error;
NETWORK
TITLE = HMI status mode
      A #statAutoMode;
      = #HMIData.stat.Automode;
NETWORK
TITLE = HMI Reset is needed
      A #reset;
      = #HMIData.stat.Reset;
NETWORK
TITLE = Reset HMI cmd aut
      A #HMIData.cmd.Auto;
```

```
    R #HMIData.cmd.Auto;
NETWORK
TITLE = Reset HMI cmd man
    A #HMIData.cmd.Man;
    R #HMIData.cmd.Man;
END_FUNCTION_BLOCK
```

Valve

The ValveOnOff function block controls an on-off valve with feedback:

```
TYPE "typeHMI_Valve"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManOpen : Bool;
      ManClose : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE
```

```
FUNCTION_BLOCK "ValveOnOff"
TITLE = ValveOnOff
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
// Siemens / (C) Copyright 2024
// ---------------------------------------------------------------------------
// Title:           ValveOnOff
// Comment/Function: Function block to control a on off valve with feedback
// Library/Family:   NA
// Author:          Automation Dept. / John Doe / john.doe@company.com
// Target System:    S7-1500/1200
// Engineering:      TIA Portal V17
// Restrictions:     None
// Requirements:     None
// ---------------------------------------------------------------------------
// Change log table:
// Version         | Date           | Expert in charge | Changes applied
//
// 001.000.000 | 2024-06-17 | John Doe             | First released version
  VAR_INPUT
    indOpened : Bool;   // Feedback valve opened
    IndClosed : Bool;   // Feedback valve closed
    indErr : Bool;   // External Error
    indInterlock : Bool;   // 0 = interlock inactive, 1 =  interlock active
    cmdAut : Bool;   // Set in auto mode (p_trig)
    cmdAutOpen : Bool;   // Open if in auto state
    cmdReset : Bool;   // Reset error
    timeout : Time;   // Timeout for feedback
  END_VAR
```

```
  VAR_OUTPUT
    Open : Bool;   // Open command
    Close : Bool;   // Close command
    Err : Bool;   // Error
    modeAut : Bool;   // In auto state
    reset : Bool;   // Reset is required
  END_VAR

  VAR_IN_OUT
    HMIData : "typeHMI_Valve";   // Data to and from HMI
  END_VAR

  VAR
    statAutoMode : Bool;
    statFeedbackError { S7_SetPoint := 'True'} : Bool;
    statExternalError : Bool;
    statResetTrig : Bool;
    statAutTrig : Bool;
    feedbackOpen {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
    feedbackClose {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
  END_VAR



BEGIN
NETWORK
TITLE = Error check for feedback Open
    A #Open;
    AN #indOpened;
    = %L0.0;
```

```
    BLD 103;
    CALL #feedbackOpen
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
       PT                  := #timeout ,
       Q                   := #statFeedbackError
    );
    NOP 0;
NETWORK
TITLE = Error check for feedback Close
    A #Close;
    AN #IndClosed;
    = %L0.0;
    BLD 103;
    CALL #feedbackClose
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
       PT                  := #timeout ,
       Q                   := #statFeedbackError
    );
    NOP 0;
NETWORK
TITLE = External Error
    A #indErr;
    = #statExternalError;
NETWORK
TITLE = Error
    O #statExternalError;
    O #statFeedbackError;
    S #Err;
```

```
NETWORK
TITLE = Auto mode
    A(;
    A #cmdAut;
    FP #statAutTrig;
    O #HMIData.cmd.Auto;
    );
    AN #HMIData.cmd.Man;
    S #statAutoMode;
NETWORK
TITLE = Manuel mode
    A #HMIData.cmd.Man;
    R #statAutoMode;
NETWORK
TITLE = Open valve
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManOpen;
    O;
    A #statAutoMode;
    A #cmdAutOpen;
    );
    AN #Err;
    AN #indInterlock;
    = #Open;
NETWORK
TITLE = Close valve
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManClose;
```

```
    O;
    A #statAutoMode;
    AN #cmdAutOpen;
    );
    AN #Err;
    AN #indInterlock;
    R #Close;
NETWORK
TITLE = Reset Error
    A(;
    O #cmdReset;
    O #HMIData.cmd.Reset;
    );
    A #reset;
    FP #statResetTrig;
    R #Err;
NETWORK
TITLE = Status mode
    A #statAutoMode;
    = #modeAut;
NETWORK
TITLE = Reset is needed
    A #Err;
    AN #statExternalError;
    AN #statFeedbackError;
    = #reset;
NETWORK
TITLE = HMI Stat Error
    A #Err;
    = #HMIData.stat.Error;
```

```
NETWORK
TITLE = HMI status mode
    A #statAutoMode;
    = #HMIData.stat.Automode;
NETWORK
TITLE = HMI Reset is needed
    A #reset;
    = #HMIData.stat.Reset;
NETWORK
TITLE = Reset HMI cmd auto
    A #HMIData.cmd.Auto;
    R #HMIData.cmd.Auto;
NETWORK
TITLE = Reset HMI cmd man
    A #HMIData.cmd.Man;
    R #HMIData.cmd.Man;
END_FUNCTION_BLOCK
```

Heater

The HeaterOnOff function block controls an on-off heater with feedback:

```
TYPE "typeHMI_Heater"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManOn : Bool;
```

```
      ManOff : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE

FUNCTION_BLOCK "HeaterOnOff"
TITLE = HeaterOnOff
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
// Siemens / (C) Copyright 2024
// ---------------------------------------------------------------------------
// Title:           HeaterOnOff
// Comment/Function: Function block to control a on off heater with feedback
// Library/Family:   NA
// Author:          Automation Dept. / John Doe / john.doe@company.com
// Target System:    S7-1500/1200
// Engineering:      TIA Portal V17
// Restrictions:     None
// Requirements:     None
// ---------------------------------------------------------------------------
// Change log table:
// Version         | Date            | Expert in charge | Changes applied
//
```

```
// 001.000.000 | 2024-06-17 | John Doe            | First released version
  VAR_INPUT
    indOn : Bool;   // Feedback
    indErr : Bool;   // External Error
    indInterlock : Bool;   // 0 = interlock inactive, 1 =  interlock active
    cmdAut : Bool;   // Set in auto mode (p_trig)
    cmdAutOn : Bool;   // Start if in auto state
    cmdReset : Bool;   // Reset error
    timeout : Time;   // Timeout for feedback
  END_VAR

  VAR_OUTPUT
    On : Bool;   // Start command
    Err : Bool;   // Error
    modeAut : Bool;   // In auto state
    reset : Bool;   // Reset is required
  END_VAR

  VAR_IN_OUT
    HMIData : "typeHMI_Heater";   // Data to and from HMI
  END_VAR

  VAR
    statAutoMode : Bool;
    feedbackOn {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
    statFeedbackError { S7_SetPoint := 'True'} : Bool;
    statExternalError : Bool;
    statResetTrig : Bool;
    statAutTrig : Bool;
  END_VAR
```

```
BEGIN
NETWORK
TITLE = Error check for feedback On
    A #On;
    AN #indOn;
    = %L0.0;
    BLD 103;
    CALL #feedbackOn
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
      PT                   := #timeout ,
      Q                    := #statFeedbackError
    );
    NOP 0;
NETWORK
TITLE = External Error
    A #indErr;
    = #statExternalError;
NETWORK
TITLE = Error
    O #statExternalError;
    O #statFeedbackError;
    S #Err;
NETWORK
TITLE = Auto mode
    A(;
    A #cmdAut;
    FP #statAutTrig;
```

```
      O #HMIData.cmd.Auto;
      );
      AN #HMIData.cmd.Man;
      S #statAutoMode;
NETWORK
TITLE = Manuel mode
      A #HMIData.cmd.Man;
      R #statAutoMode;
NETWORK
TITLE = Start heater
      A(;
      AN #statAutoMode;
      A #HMIData.cmd.ManOn;
      O;
      A #statAutoMode;
      A #cmdAutOn;
      );
      AN #HMIData.cmd.ManOff;
      AN #Err;
      AN #indInterlock;
      = #On;
NETWORK
TITLE = Reset Error
      A(;
      O #cmdReset;
      O #HMIData.cmd.Reset;
      );
      A #reset;
      FP #statResetTrig;
      R #Err;
```

```
NETWORK
TITLE = Status mode
    A #statAutoMode;
    = #modeAut;
NETWORK
TITLE = Reset is needed
    A #Err;
    AN #statExternalError;
    AN #statFeedbackError;
    = #reset;
NETWORK
TITLE = HMI Stat Error
    A #Err;
    = #HMIData.stat.Error;
NETWORK
TITLE = HMI status mode
    A #statAutoMode;
    = #HMIData.stat.Automode;
NETWORK
TITLE = HMI Reset is needed
    A #reset;
    = #HMIData.stat.Reset;
NETWORK
TITLE = Reset HMI cmd aut
    A #HMIData.cmd.Auto;
    R #HMIData.cmd.Auto;
NETWORK
TITLE = Reset HMI cmd man
    A #HMIData.cmd.Man;
    R #HMIData.cmd.Man;
```

```
END_FUNCTION_BLOCK
```

Analogue sensor

The AnalogSensor function block processes analog sensor values, this function block cannot be converted to LAD so do not use the logic normally when creating logic, it should only be used in this case:

```
TYPE "typeHMI_Sensor"
VERSION : 0.1
  STRUCT
    ScaledValue : Real;
  END_STRUCT;

END_TYPE

FUNCTION_BLOCK "AnalogSensor"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  VAR_INPUT
    rawValue : Int;   // Raw value from PLC
    scaleMax : Real;   // Engineering unit max
    scaleMin : Real;   // Engineering unit min
  END_VAR

  VAR_OUTPUT
    scaledValue : Real;   // Scaled value
  END_VAR

  VAR_IN_OUT
```

```
    HMIdata : "typeHMI_Sensor";   // Data to and from HMI
  END_VAR



BEGIN
NETWORK
TITLE =
    L #rawValue           ;// Load the raw analog input value
    ITD           ;// Convert integer to double integer
    DTR           ;// Convert double integer to real number

    L 27648.0      ;// Load the maximum raw value for analog input
    /R            ;// Divide the raw value by 27648.0 to normalize (0.0 to 1.0)

    L #scaleMax         ;// Load the maximum value of the scaled range
    L #scaleMin         ;// Load the minimum value of the scaled range
    -R            ;// Subtract the minimum value from the maximum value
    *R            ;// Multiply the normalized value by the range (max - min)

    L #scaleMin         ;// Load the minimum value of the scaled range
    +R            ;// Add the minimum value to complete the scaling

    T #scaledValue      ;// Store the result in the output variable
    T #HMIdata.ScaledValue;// Move value to HMI data

END_FUNCTION_BLOCK
```

PLC tags

In the section for “Multiple functions and instances” I have used “PLC tags”. The meaning of PLC tags is to connect them with the hardware I/O addresses to a symbolic name, so it is easier to read and understand

the input and output of PLC instead of absolute addresses such as I0.0 and Q0.1 and so on. Therefore, if you encounter a I/O list, then create a excel file to import the tags to TIA portal, here is an example and the PLC tags that can be imported to TIA portal and are used in the following section as an application example, remember to have all ROWS:

Name	Path	Data Type	Logical Address		Comment	Hmi Visible	Hmi Accessible	Hmi Writeable	Typeobject ID	Version ID
MA01_On	Tag table_1	Bool	%I0.0		True	True	True	True		
EB01_On	Tag table_1	Bool	%I0.1		True	True	True	True		
MB01_Opened	Tag table_1	Bool	%I0.2			True	True	True		
MB01_Closed	Tag table_1	Bool	%I0.3		True	True	True	True		
MA01_cmdOn	Tag table_1	Bool	%Q0.0		True	True	True	True		
EB01_cmdOn	Tag table_1	Bool	%Q0.1		True	True	True	True		
MB01_cmdOpen	Tag table_1	Bool	%Q0.2			True	True	True		
MB01_cmdClose	Tag table_1	Bool	%Q0.3			True	True	True		
S1_Start	Tag table_1	Bool	%I0.4		True	True	True	True		
S2_Stop	Tag table_1	Bool	%I0.5		True	True	True	True		
S3_EmgStop	Tag table_1	Bool	%I0.6		True	True	True	True		
BT01_TempTank	Tag table_1	Int	%IW100			True	True	True		
BL01_TankHigh	Tag table_1	Bool	%I0.7			True	True	True		
BL02_TankLow	Tag table_1	Bool	%I1.0			True	True	True		
FS_FillingReady	Tag table_1	Bool	%I1.1			True	True	True		

Multiple functions and instances

Description

It is a good idea to capsulate functions inside other functions to handle program and make it easier for other to use and re-use functions.

For this example, I have used multiple instance function inside TIA portal. So, by calling other functions and declaring the instances DB in the call functions it can capsulate the machine and its function and sub functions. Then use locale static or temporary variable to connect outputs from let's say a control block to a component block and vice versa.

Calling functions with multiple bool input and using global PLC tags and global DB

When handling a function with multiple bool inputs add a local memory to store it followed by BLD 103 after each bool input for setting the parameter of the function call. When using global tags like PLC tags use "" for example "myPLCtag" and global DB with "" and a "." for the tag inside the global DB for example "MyGlobalDB".MyfirstTag, see this example:

```
TITLE = MA01 pump
    A "MA01_On";
    = %L1.0;
    BLD 103;
    A #PumpOn;
    = %L1.1;
    BLD 103;
    CALL  #intsMotorOnOff_MA01
      indOn       :=%L1.0
      indErr      :=
      indInterlock :=
      cmdAut      :=
      cmdAutOn    :=%L1.1
      cmdReset    :=
      timeout     :=
      On          :="MA01_cmdOn"
      Err         :=
      modeAut     :=
```

```
      reset       :=
      HMIData     :="HMI".MA01
    );
    NOP 0;
```

In this example the “MA01_On” is feedback from a PLC tag assign to a PLC I/O card. And the #PumpOn is a local tag from a control block to start the motor when the motor is in auto mode. And because this function blocks have more than one bool input on its parameter, we have to use local memory %L0.0 and BLD 103 for the code to convert to LAD. And as you can see the output from the block is assigned to a PLC tag to start the motor. It is okay to leave out some of the parameter assignment if they are not relevant for the project, in this case we don’t have feedback for Err, Interlock and Reset and therefore not in the calling code, even though the block itself have these inputs parameters. The data to and from the function block to the HMI is handle with a INOUT parameter and is assigned to a global DB that contains the type of a motor.

Application example of calling multiple self-created functions that is called inside a main function.

Connect relevant PLC tags to the blocks parameter as showed in this example. The PLC tags are from the previous section “PLC tags” that are used. Then you can also add a global DB for interface with HMI. Therefore, also declare a PLC data type so it is easier to maintain. Please see the example provided. This code is able to be converted to LAD and it is important to follow the example provided or use as much as you can when dealing with a new situation to keep the same structure and syntaxes. Keep in mind as with the timers to use %L and BLD103 and NOP 0 at the start and beginning of calling sub functions blocks:

```
FUNCTION_BLOCK "HeatingTankMainFB"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  VAR
    instSekvensHeatingTank : "SeqHeatingTank";
    PumpOn { S7_SetPoint := 'True'} : Bool;
    HeaterOn : Bool;
    ValveOn : Bool;
    Temperature : Real;
    intsMotorOnOff_MA01 : "MotorOnOff";
    intsHeaterOnOff_EB01 : "HeaterOnOff";
    instValveOnOff_MB01 : "ValveOnOff";
    instAnalogSensor_BT01 : "AnalogSensor";
```

```
  END_VAR


BEGIN
NETWORK
TITLE = Control blok for seqence
    A(;
    O "S1_Start";
    O "HMI".Start;
    );
    = %L1.0;
    BLD 103;
    A(;
    O "S2_Stop";
    O "HMI".Stop;
    );
    = %L1.1;
    BLD 103;
    A "S3_EmgStop";
    = %L1.2;
    BLD 103;
    A "BL01_TankHigh";
    = %L1.3;
    BLD 103;
    A "BL02_TankLow";
    = %L1.4;
    BLD 103;
    A "FS_FillingReady";
    = %L1.5;
    BLD 103;
    A "MA01_On";
```

```
= %L1.6;
BLD 103;
A "EB01_On";
= %L1.7;
BLD 103;
A "MB01_Opened";
= %L2.0;
BLD 103;
A "MB01_Closed";
= %L2.1;
BLD 103;
CALL #instSekvensHeatingTank
(  Start                  := %L1.0 ,
   Stop                   := %L1.1 ,
   EmgStop                := %L1.2 ,
   temperatureSensor      := #Temperature ,
   levelSensorHigh        := %L1.3 ,
   levelSensorLow         := %L1.4 ,
   FillingStationReady    := %L1.5 ,
   pumpRunning            := %L1.6 ,
   heaterOn               := %L1.7 ,
   valveOpened            := %L2.0 ,
   valveClosed            := %L2.1 ,
   setTemperature         := 50.0 ,
   deadbandTemperature    := 2.0 ,
   cmdPumpOn              := #PumpOn ,
   cmdHeaterOn            := #HeaterOn ,
   cmdValveOn             := #ValveOn ,
   stateStep              := "HMI"."HeatingTank Step"
);
```

```
    NOP 0;
NETWORK
TITLE = MA01 pump
    A "MA01_On";
    = %L1.0;
    BLD 103;
    A #PumpOn;
    = %L1.1;
    BLD 103;
    CALL #intsMotorOnOff_MA01
    (  indOn                  := %L1.0 ,
      cmdAutOn                 := %L1.1 ,
      On                     := "MA01_cmdOn" ,
      HMIData                  := "HMI".MA01
    );
    NOP 0;
NETWORK
TITLE = EB01 Heater
    A "EB01_On";
    = %L1.0;
    BLD 103;
    A #HeaterOn;
    = %L1.1;
    BLD 103;
    CALL #intsHeaterOnOff_EB01
    (  indOn                  := %L1.0 ,
      cmdAutOn                 := %L1.1 ,
      On                     := "EB01_cmdOn" ,
      HMIData                  := "HMI".EB01
    );
```

```
    NOP 0;
NETWORK
TITLE =
    A "MB01_Opened";
    = %L1.0;
    BLD 103;
    A "MB01_Closed";
    = %L1.1;
    BLD 103;
    A #ValveOn;
    = %L1.2;
    BLD 103;
    CALL #instValveOnOff_MB01
    (  indOpened              := %L1.0 ,
      IndClosed               := %L1.1 ,
      cmdAutOpen              := %L1.2 ,
      Open                    := "MB01_cmdOpen" ,
      Close                   := "MB01_cmdClose" ,
      HMIData                 := "HMI".MB01
    );
    NOP 0;
NETWORK
TITLE = BT01 Temperature
    CALL #instAnalogSensor_BT01
    (  rawValue               := "BT01_TempTank" ,
      scaleMax                := 100.0 ,
      scaleMin                := 0.0 ,
      scaledValue             := #Temperature ,
      HMIdata                 := "HMI".BT01
    );
```

```
    NOP 0;
END_FUNCTION_BLOCK
```

Global DB to handle HMI data

When you have created the functions blocks and the necessary main block you can connect data to the blocks parameters as showed in the previous section. But for the HMI or SCADA system to connect with the data in and out of the function block create a global DB containing the types of component such as motors, valve, sensor and so on and use the type for this. You can also create an HMI start and stop variable and so on, se this example:

```
TYPE "typeHMI_Motor"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManStart : Bool;
      ManStop : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE

TYPE "typeHMI_Heater"
VERSION : 0.1
```

```
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManOn : Bool;
      ManOff : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE

TYPE "typeHMI_Valve"
VERSION : 0.1
  STRUCT
    cmd : Struct
      Auto : Bool;
      Man : Bool;
      ManOpen : Bool;
      ManClose : Bool;
      Reset : Bool;
    END_STRUCT;
    stat : Struct
      Automode : Bool;
      Error : Bool;
```

```
      Reset : Bool;
    END_STRUCT;
  END_STRUCT;

END_TYPE

TYPE "typeHMI_Sensor"
VERSION : 0.1
  STRUCT
    ScaledValue : Real;
  END_STRUCT;

END_TYPE

DATA_BLOCK "HMI"
{ S7_Optimized_Access := 'FALSE' }
VERSION : 0.1
  STRUCT
    Start : Bool;
    Stop : Bool;
    MA01 { S7_SetPoint := 'False'} : "typeHMI_Motor";
    EB01 : "typeHMI_Heater";
    MB01 : "typeHMI_Valve";
    BT01 : "typeHMI_Sensor";
    "HeatingTank Step" { S7_SetPoint := 'True'} : Int;
  END_STRUCT;


BEGIN
```

```
END_DATA_BLOCK
```

If the datatype is already defined elsewhere, you can exclude it from the AWL file for the global DB. But you must be sure that you either have created a separated AWL file for types or included the types inside the functions block AWL file

Conclusion

You now have a base idea of how to create simple sub function of controlling components and have a good idea to create a sequence control in LAD using move compare method as well with capsulation function with multiple functions and instances inside a main function block, and how to assign and create PLC tags and global DB for handle hardware IN/OUT and HMI data IN/OUT.

Bilag 3: Eksempel på chat med chat GPT STL

https://chatgpt.com/share/f40e7c7f-2010-4a3d-8d53-762b65fca1e9

Bilag 4: Eksempel AWL for en pumpe

```
FUNCTION_BLOCK "PumpControl"

TITLE = PumpControl

{ S7_Optimized_Access := 'FALSE' }

VERSION : 0.1

  VAR_INPUT

    indOn : Bool;   // Feedback

    indErr : Bool;   // External Error

    indInterlock : Bool;   // Interlock status

    cmdAut : Bool;   // Set in auto mode

    cmdAutOn : Bool;   // Start if in auto state

    cmdReset : Bool;   // Reset error

    timeout : Time;   // Timeout for feedback

  END_VAR


  VAR_OUTPUT

    On : Bool;   // Start command

    Err : Bool;   // Error

    modeAut : Bool;   // In auto state

    reset : Bool;   // Reset is required
```

```
  END_VAR

  VAR_IN_OUT
    HMIData : "typeHMI_Component";   // Data to and from HMI
  END_VAR

  VAR
    statAutoMode : Bool;
    feedbackOn {InstructionName := 'TON'; LibVersion := '1.0'} : TON;
    statFeedbackError { S7_SetPoint := 'True'} : Bool;
    statExternalError : Bool;
    statResetTrig : Bool;
    statAutTrig : Bool;
  END_VAR


BEGIN
NETWORK
TITLE = Error check for feedback On
    A #On;
    AN #indOn;
    = %L0.0;
    BLD 103;
    CALL #feedbackOn
    {time_type := 'Time'}
    (  IN                  := %L0.0 ,
      PT                   := #timeout ,
      Q                    := #statFeedbackError
    );
    NOP 0;
```

```
NETWORK
TITLE = External Error
    A #indErr;
    = #statExternalError;
NETWORK
TITLE = Error
    O #statExternalError;
    O #statFeedbackError;
    S #Err;
NETWORK
TITLE = Auto mode
    A(;
    A #cmdAut;
    FP #statAutTrig;
    O #HMIData.cmd.Auto;
    );
    AN #HMIData.cmd.Man;
    S #statAutoMode;
NETWORK
TITLE = Manual mode
    A #HMIData.cmd.Man;
    R #statAutoMode;
NETWORK
TITLE = Start pump
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManStart;
    O;
    A #statAutoMode;
    A #cmdAutOn;
```

```
    );
    AN #Err;
    AN #indInterlock;
    = #On;
NETWORK
TITLE = Stop pump
    A(;
    AN #statAutoMode;
    A #HMIData.cmd.ManStop;
    O;
    A #statAutoMode;
    AN #cmdAutOn;
    );
    AN #Err;
    AN #indInterlock;
    R #On;
NETWORK
TITLE = Reset Error
    A(;
    O #cmdReset;
    O #HMIData.cmd.Reset;
    );
    A #reset;
    FP #statResetTrig;
    R #Err;
NETWORK
TITLE = Status mode
    A #statAutoMode;
    = #modeAut;
NETWORK
```

```
TITLE = Reset is needed
    A #Err;
    AN #statExternalError;
    AN #statFeedbackError;
    = #reset;
NETWORK
TITLE = HMI Stat Error
    A #Err;
    = #HMIData.stat.Error;
NETWORK
TITLE = HMI status mode
    A #statAutoMode;
    = #HMIData.stat.Automode;
NETWORK
TITLE = HMI Reset is needed
    A #reset;
    = #HMIData.stat.Reset;
NETWORK
TITLE = Reset HMI cmd auto
    A #HMIData.cmd.Auto;
    R #HMIData.cmd.Auto;
NETWORK
TITLE = Reset HMI cmd manual
    A #HMIData.cmd.Man;
    R #HMIData.cmd.Man;
END_FUNCTION_BLOCK
```

Bilag 5: Eksempel på XML fra TIA

```
<Document>
<Engineering version="V17"/>
<DocumentInfo>
<Created>2024-06-24T12:27:15.2522709Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
```

```
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>TIA Portal Openness</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</OptionPackage>
<OptionPackage>
<DisplayName>TIA Portal Version Control Interface</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V17 Update 6</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>WinCC Professional</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>SIMATIC Visualization Architect</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<SW.Blocks.FB ID="0">
<AttributeList>
<Interface>
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v5">
<Section Name="Input">
<Member Name="indOn" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Feedback</MultiLanguageText>
</Comment>
</Member>
<Member Name="indErr" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">External Error</MultiLanguageText>
</Comment>
</Member>
<Member Name="indInterlock" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Interlock status</MultiLanguageText>
</Comment>
</Member>
<Member Name="cmdAut" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Set in auto mode</MultiLanguageText>
</Comment>
</Member>
<Member Name="cmdAutOn" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Start if in auto state</MultiLanguageText>
</Comment>
```

```
</Member>
<Member Name="cmdReset" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Reset error</MultiLanguageText>
</Comment>
</Member>
<Member Name="timeout" Datatype="Time">
<Comment>
<MultiLanguageText Lang="en-US">Timeout for feedback</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="Output">
<Member Name="On" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Start command</MultiLanguageText>
</Comment>
</Member>
<Member Name="Err" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Error</MultiLanguageText>
</Comment>
</Member>
<Member Name="modeAut" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">In auto state</MultiLanguageText>
</Comment>
</Member>
<Member Name="reset" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Reset is required</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="InOut">
<Member Name="HMIData" Datatype=""typeHMI_Component"">
<Comment>
<MultiLanguageText Lang="en-US">Data to and from HMI</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="Static">
<Member Name="statAutoMode" Datatype="Bool"/>
<Member Name="feedbackOn" Datatype="TON" Version="1.0">
<AttributeList>
<BooleanAttribute Name="SetPoint" SystemDefined="true">true</BooleanAttribute>
</AttributeList>
</Member>
<Member Name="statFeedbackError" Datatype="Bool">
<AttributeList>
<BooleanAttribute Name="SetPoint" SystemDefined="true">true</BooleanAttribute>
</AttributeList>
</Member>
<Member Name="statExternalError" Datatype="Bool"/>
<Member Name="statResetTrig" Datatype="Bool"/>
<Member Name="statAutTrig" Datatype="Bool"/>
</Section>
<Section Name="Temp"/>
<Section Name="Constant"/>
```

```
</Sections>
</Interface>
<Name>PumpControl</Name>
<Number>1</Number>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<SW.Blocks.CompileUnit ID="3" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="On"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="indOn"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="timeout"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
<Component Name="statFeedbackError"/>
</Symbol>
</Access>
<Part Name="Contact" UId="25"/>
<Part Name="Contact" UId="26">
<Negated Name="operand"/>
</Part>
<Part Name="TON" Version="1.0" UId="27">
<Instance Scope="LocalVariable" UId="28">
<Component Name="feedbackOn"/>
</Instance>
<TemplateValue Name="time_type" Type="Type">Time</TemplateValue>
</Part>
<Part Name="Coil" UId="29"/>
</Parts>
<Wires>
<Wire UId="31">
<IdentCon UId="21"/>
<NameCon UId="25" Name="operand"/>
```

```
</Wire>
<Wire UId="32">
<IdentCon UId="22"/>
<NameCon UId="26" Name="operand"/>
</Wire>
<Wire UId="33">
<IdentCon UId="23"/>
<NameCon UId="27" Name="PT"/>
</Wire>
<Wire UId="34">
<NameCon UId="27" Name="ET"/>
<OpenCon UId="30"/>
</Wire>
<Wire UId="35">
<NameCon UId="27" Name="Q"/>
<NameCon UId="29" Name="in"/>
</Wire>
<Wire UId="36">
<IdentCon UId="24"/>
<NameCon UId="29" Name="operand"/>
</Wire>
<Wire UId="37">
<NameCon UId="25" Name="out"/>
<NameCon UId="26" Name="in"/>
</Wire>
<Wire UId="38">
<NameCon UId="26" Name="out"/>
<NameCon UId="27" Name="IN"/>
</Wire>
<Wire UId="39">
<Powerrail/>
<NameCon UId="25" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="4" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="5" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="6" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="7" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Error check for feedback On</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
```

```
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="8" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="indErr"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="statExternalError"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="Coil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="9" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="A" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="B" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="C" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
```

<Text>External Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="D" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="statExternalError"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="statFeedbackError"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Part Name="Contact" UId="24"/>
<Part Name="Contact" UId="25"/>
<Part Name="O" UId="26">
<TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
</Part>
<Part Name="SCoil" UId="27"/>
</Parts>
<Wires>
<Wire UId="28">
<IdentCon UId="21"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="29">
<IdentCon UId="22"/>
<NameCon UId="25" Name="operand"/>
</Wire>
<Wire UId="30">
<NameCon UId="26" Name="out"/>
<NameCon UId="27" Name="in"/>
</Wire>
<Wire UId="31">
<IdentCon UId="23"/>
<NameCon UId="27" Name="operand"/>
</Wire>
<Wire UId="32">
<NameCon UId="24" Name="out"/>
<NameCon UId="26" Name="in1"/>
</Wire>
<Wire UId="33">
<NameCon UId="25" Name="out"/>
<NameCon UId="26" Name="in2"/>

```
</Wire>
<Wire UId="34">
<Powerrail/>
<NameCon UId="24" Name="in"/>
<NameCon UId="25" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="E" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="10" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="11" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="12" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="cmdAut"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="statAutTrig"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Auto"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
```

```
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Man"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="25">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Part Name="Contact" UId="26"/>
<Part Name="PBox" UId="27"/>
<Part Name="Contact" UId="28"/>
<Part Name="O" UId="29">
<TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
</Part>
<Part Name="Contact" UId="30">
<Negated Name="operand"/>
</Part>
<Part Name="SCoil" UId="31"/>
</Parts>
<Wires>
<Wire UId="32">
<IdentCon UId="21"/>
<NameCon UId="26" Name="operand"/>
</Wire>
<Wire UId="33">
<IdentCon UId="22"/>
<NameCon UId="27" Name="bit"/>
</Wire>
<Wire UId="34">
<NameCon UId="27" Name="out"/>
<NameCon UId="29" Name="in1"/>
</Wire>
<Wire UId="35">
<IdentCon UId="23"/>
<NameCon UId="28" Name="operand"/>
</Wire>
<Wire UId="36">
<NameCon UId="29" Name="out"/>
<NameCon UId="30" Name="in"/>
</Wire>
<Wire UId="37">
<IdentCon UId="24"/>
<NameCon UId="30" Name="operand"/>
</Wire>
<Wire UId="38">
<IdentCon UId="25"/>
<NameCon UId="31" Name="operand"/>
</Wire>
<Wire UId="39">
<NameCon UId="26" Name="out"/>
<NameCon UId="27" Name="in"/>
</Wire>
<Wire UId="40">
<NameCon UId="28" Name="out"/>
<NameCon UId="29" Name="in2"/>
</Wire>
<Wire UId="41">
```

```
<NameCon UId="30" Name="out"/>
<NameCon UId="31" Name="in"/>
</Wire>
<Wire UId="42">
<Powerrail/>
<NameCon UId="26" Name="in"/>
<NameCon UId="28" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="13" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="14" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="15" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="16" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Auto mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="17" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Man"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="RCoil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
```

```
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="18" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="19" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="1A" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="1B" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Manual mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="1C" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="ManStart"/>
```

```
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
<Component Name="cmdAutOn"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="25">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="26">
<Symbol>
<Component Name="indInterlock"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="27">
<Symbol>
<Component Name="On"/>
</Symbol>
</Access>
<Part Name="Contact" UId="28">
<Negated Name="operand"/>
</Part>
<Part Name="Contact" UId="29"/>
<Part Name="Contact" UId="30"/>
<Part Name="Contact" UId="31"/>
<Part Name="O" UId="32">
<TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
</Part>
<Part Name="Contact" UId="33">
<Negated Name="operand"/>
</Part>
<Part Name="Contact" UId="34">
<Negated Name="operand"/>
</Part>
<Part Name="Coil" UId="35"/>
</Parts>
<Wires>
<Wire UId="36">
<IdentCon UId="21"/>
<NameCon UId="28" Name="operand"/>
</Wire>
<Wire UId="37">
<IdentCon UId="22"/>
<NameCon UId="29" Name="operand"/>
</Wire>
<Wire UId="38">
<IdentCon UId="23"/>
<NameCon UId="30" Name="operand"/>
</Wire>
<Wire UId="39">
<IdentCon UId="24"/>
```

<NameCon UId="31" Name="operand"/>
</Wire>
<Wire UId="40">
<NameCon UId="32" Name="out"/>
<NameCon UId="33" Name="in"/>
</Wire>
<Wire UId="41">
<IdentCon UId="25"/>
<NameCon UId="33" Name="operand"/>
</Wire>
<Wire UId="42">
<IdentCon UId="26"/>
<NameCon UId="34" Name="operand"/>
</Wire>
<Wire UId="43">
<IdentCon UId="27"/>
<NameCon UId="35" Name="operand"/>
</Wire>
<Wire UId="44">
<NameCon UId="28" Name="out"/>
<NameCon UId="29" Name="in"/>
</Wire>
<Wire UId="45">
<NameCon UId="29" Name="out"/>
<NameCon UId="32" Name="in1"/>
</Wire>
<Wire UId="46">
<NameCon UId="30" Name="out"/>
<NameCon UId="31" Name="in"/>
</Wire>
<Wire UId="47">
<NameCon UId="31" Name="out"/>
<NameCon UId="32" Name="in2"/>
</Wire>
<Wire UId="48">
<NameCon UId="33" Name="out"/>
<NameCon UId="34" Name="in"/>
</Wire>
<Wire UId="49">
<NameCon UId="34" Name="out"/>
<NameCon UId="35" Name="in"/>
</Wire>
<Wire UId="50">
<Powerrail/>
<NameCon UId="28" Name="in"/>
<NameCon UId="30" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1D" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="1E" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>

```
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="1F" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="20" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Start pump</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="21" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="ManStop"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
<Component Name="cmdAutOn"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="25">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="26">
<Symbol>
<Component Name="indInterlock"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="27">
<Symbol>
<Component Name="On"/>
</Symbol>
</Access>
```

```
<Part Name="Contact" UId="28">
<Negated Name="operand"/>
</Part>
<Part Name="Contact" UId="29"/>
<Part Name="Contact" UId="30"/>
<Part Name="Contact" UId="31">
<Negated Name="operand"/>
</Part>
<Part Name="O" UId="32">
<TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
</Part>
<Part Name="Contact" UId="33">
<Negated Name="operand"/>
</Part>
<Part Name="Contact" UId="34">
<Negated Name="operand"/>
</Part>
<Part Name="RCoil" UId="35"/>
</Parts>
<Wires>
<Wire UId="36">
<IdentCon UId="21"/>
<NameCon UId="28" Name="operand"/>
</Wire>
<Wire UId="37">
<IdentCon UId="22"/>
<NameCon UId="29" Name="operand"/>
</Wire>
<Wire UId="38">
<IdentCon UId="23"/>
<NameCon UId="30" Name="operand"/>
</Wire>
<Wire UId="39">
<IdentCon UId="24"/>
<NameCon UId="31" Name="operand"/>
</Wire>
<Wire UId="40">
<NameCon UId="32" Name="out"/>
<NameCon UId="33" Name="in"/>
</Wire>
<Wire UId="41">
<IdentCon UId="25"/>
<NameCon UId="33" Name="operand"/>
</Wire>
<Wire UId="42">
<IdentCon UId="26"/>
<NameCon UId="34" Name="operand"/>
</Wire>
<Wire UId="43">
<IdentCon UId="27"/>
<NameCon UId="35" Name="operand"/>
</Wire>
<Wire UId="44">
<NameCon UId="28" Name="out"/>
<NameCon UId="29" Name="in"/>
</Wire>
<Wire UId="45">
<NameCon UId="29" Name="out"/>
<NameCon UId="32" Name="in1"/>
```

```
</Wire>
<Wire UId="46">
<NameCon UId="30" Name="out"/>
<NameCon UId="31" Name="in"/>
</Wire>
<Wire UId="47">
<NameCon UId="31" Name="out"/>
<NameCon UId="32" Name="in2"/>
</Wire>
<Wire UId="48">
<NameCon UId="33" Name="out"/>
<NameCon UId="34" Name="in"/>
</Wire>
<Wire UId="49">
<NameCon UId="34" Name="out"/>
<NameCon UId="35" Name="in"/>
</Wire>
<Wire UId="50">
<Powerrail/>
<NameCon UId="28" Name="in"/>
<NameCon UId="30" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="22" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="23" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="24" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="25" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Stop pump</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="26" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="cmdReset"/>
```

```
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Reset"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="reset"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
<Component Name="statResetTrig"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="25">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Part Name="Contact" UId="26"/>
<Part Name="Contact" UId="27"/>
<Part Name="O" UId="28">
<TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
</Part>
<Part Name="Contact" UId="29"/>
<Part Name="PBox" UId="30"/>
<Part Name="RCoil" UId="31"/>
</Parts>
<Wires>
<Wire UId="32">
<IdentCon UId="21"/>
<NameCon UId="26" Name="operand"/>
</Wire>
<Wire UId="33">
<IdentCon UId="22"/>
<NameCon UId="27" Name="operand"/>
</Wire>
<Wire UId="34">
<NameCon UId="28" Name="out"/>
<NameCon UId="29" Name="in"/>
</Wire>
<Wire UId="35">
<IdentCon UId="23"/>
<NameCon UId="29" Name="operand"/>
</Wire>
<Wire UId="36">
<IdentCon UId="24"/>
<NameCon UId="30" Name="bit"/>
</Wire>
<Wire UId="37">
<NameCon UId="30" Name="out"/>
<NameCon UId="31" Name="in"/>
</Wire>
<Wire UId="38">
```

```
<IdentCon UId="25"/>
<NameCon UId="31" Name="operand"/>
</Wire>
<Wire UId="39">
<NameCon UId="26" Name="out"/>
<NameCon UId="28" Name="in1"/>
</Wire>
<Wire UId="40">
<NameCon UId="27" Name="out"/>
<NameCon UId="28" Name="in2"/>
</Wire>
<Wire UId="41">
<NameCon UId="29" Name="out"/>
<NameCon UId="30" Name="in"/>
</Wire>
<Wire UId="42">
<Powerrail/>
<NameCon UId="26" Name="in"/>
<NameCon UId="27" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="27" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="28" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="29" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="2A" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="2B" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
```

```
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="modeAut"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="Coil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="2C" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2D" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="2E" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="2F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Status mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="30" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
```

```
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="statExternalError"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="23">
<Symbol>
<Component Name="statFeedbackError"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="24">
<Symbol>
<Component Name="reset"/>
</Symbol>
</Access>
<Part Name="Contact" UId="25"/>
<Part Name="Contact" UId="26">
<Negated Name="operand"/>
</Part>
<Part Name="Contact" UId="27">
<Negated Name="operand"/>
</Part>
<Part Name="Coil" UId="28"/>
</Parts>
<Wires>
<Wire UId="29">
<IdentCon UId="21"/>
<NameCon UId="25" Name="operand"/>
</Wire>
<Wire UId="30">
<IdentCon UId="22"/>
<NameCon UId="26" Name="operand"/>
</Wire>
<Wire UId="31">
<IdentCon UId="23"/>
<NameCon UId="27" Name="operand"/>
</Wire>
<Wire UId="32">
<IdentCon UId="24"/>
<NameCon UId="28" Name="operand"/>
</Wire>
<Wire UId="33">
<NameCon UId="25" Name="out"/>
<NameCon UId="26" Name="in"/>
</Wire>
<Wire UId="34">
<NameCon UId="26" Name="out"/>
<NameCon UId="27" Name="in"/>
</Wire>
<Wire UId="35">
<NameCon UId="27" Name="out"/>
<NameCon UId="28" Name="in"/>
</Wire>
<Wire UId="36">
```

<Powerrail/>
<NameCon UId="25" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="31" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="32" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="33" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="34" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset is needed</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="35" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="Err"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="stat"/>
<Component Name="Error"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="Coil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>

```
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="36" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="37" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="38" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="39" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI Stat Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="3A" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="statAutoMode"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="stat"/>
<Component Name="Automode"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="Coil" UId="24"/>
</Parts>
```

```
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="3B" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="3C" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3D" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="3E" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI status mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="3F" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="reset"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
```

<Component Name="stat"/>
<Component Name="Reset"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="Coil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="40" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="41" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="42" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="43" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI Reset is needed</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="44" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">

<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Auto"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Auto"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="RCoil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="45" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="46" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="47" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="48" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset HMI cmd auto</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>

```
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="49" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNe
t/v4">
<Parts>
<Access Scope="LocalVariable" UId="21">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Man"/>
</Symbol>
</Access>
<Access Scope="LocalVariable" UId="22">
<Symbol>
<Component Name="HMIData"/>
<Component Name="cmd"/>
<Component Name="Man"/>
</Symbol>
</Access>
<Part Name="Contact" UId="23"/>
<Part Name="RCoil" UId="24"/>
</Parts>
<Wires>
<Wire UId="25">
<IdentCon UId="21"/>
<NameCon UId="23" Name="operand"/>
</Wire>
<Wire UId="26">
<IdentCon UId="22"/>
<NameCon UId="24" Name="operand"/>
</Wire>
<Wire UId="27">
<NameCon UId="23" Name="out"/>
<NameCon UId="24" Name="in"/>
</Wire>
<Wire UId="28">
<Powerrail/>
<NameCon UId="23" Name="in"/>
</Wire>
</Wires>
</FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="4A" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="4B" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="4C" CompositionName="Title">
```

```
<ObjectList>
<MultilingualTextItem ID="4D" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset HMI cmd manual</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<MultilingualText ID="4E" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>PumpControl</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.FB>
</Document>
```

Bilag 6: Eksempel på modificeret XML fra chat GPT

```
<Document xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v5" xml
ns:NetworkSource="http://www.siemens.com/automation/Openness/SW/NetworkSource/Fl
gNet/v4">
<Engineering version="V17"/>
<DocumentInfo>
<Created>2024-06-24T12:27:15.2522709Z</Created>
<ExportSetting>None</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>TIA Portal Openness</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</OptionPackage>
<OptionPackage>
<DisplayName>TIA Portal Version Control Interface</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>STEP 7 Professional</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>STEP 7 Safety</DisplayName>
<DisplayVersion>V17 Update 6</DisplayVersion>
</OptionPackage>
<Product>
<DisplayName>WinCC Professional</DisplayName>
<DisplayVersion>V17 Update 7</DisplayVersion>
```

```
</Product>
<OptionPackage>
<DisplayName>SIMATIC Visualization Architect</DisplayName>
<DisplayVersion>V17</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<SW.Blocks.FB ID="0">
<AttributeList>
<Interface>
<Sections>
<Section Name="Input">
<Member Name="indOn" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Feedback</MultiLanguageText>
</Comment>
</Member>
<Member Name="indErr" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">External Error</MultiLanguageText>
</Comment>
</Member>
<Member Name="indInterlock" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Interlock status</MultiLanguageText>
</Comment>
</Member>
<Member Name="cmdAut" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Set in auto mode</MultiLanguageText>
</Comment>
</Member>
<Member Name="cmdAutOn" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Start if in auto state</MultiLanguageText>
</Comment>
</Member>
<Member Name="cmdReset" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Reset error</MultiLanguageText>
</Comment>
</Member>
<Member Name="timeout" Datatype="Time">
<Comment>
<MultiLanguageText Lang="en-US">Timeout for feedback</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="Output">
<Member Name="On" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Start command</MultiLanguageText>
</Comment>
</Member>
<Member Name="Err" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Error</MultiLanguageText>
</Comment>
</Member>
```

```
<Member Name="modeAut" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">In auto state</MultiLanguageText>
</Comment>
</Member>
<Member Name="reset" Datatype="Bool">
<Comment>
<MultiLanguageText Lang="en-US">Reset is required</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="InOut">
<Member Name="HMIData" Datatype=""typeHMI_Component"">
<Comment>
<MultiLanguageText Lang="en-US">Data to and from HMI</MultiLanguageText>
</Comment>
</Member>
</Section>
<Section Name="Static">
<Member Name="statAutoMode" Datatype="Bool"/>
<Member Name="feedbackOn" Datatype="TON" Version="1.0">
<AttributeList>
<BooleanAttribute Name="SetPoint" SystemDefined="true">true</BooleanAttribute>
</AttributeList>
</Member>
<Member Name="statFeedbackError" Datatype="Bool">
<AttributeList>
<BooleanAttribute Name="SetPoint" SystemDefined="true">true</BooleanAttribute>
</AttributeList>
</Member>
<Member Name="statExternalError" Datatype="Bool"/>
<Member Name="statResetTrig" Datatype="Bool"/>
<Member Name="statAutTrig" Datatype="Bool"/>
</Section>
<Section Name="Temp"/>
<Section Name="Constant"/>
</Sections>
</Interface>
<Name>PumpControl</Name>
<Number>1</Number>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<SW.Blocks.CompileUnit ID="3" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
```

```
<NetworkSource:Symbol>
<NetworkSource:Component Name="On"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="indOn"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="timeout"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statFeedbackError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="25"/>
<NetworkSource:Part Name="Contact" UId="26">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="TON" Version="1.0" UId="27">
<NetworkSource:Instance Scope="LocalVariable" UId="28">
<NetworkSource:Component Name="feedbackOn"/>
</NetworkSource:Instance>
<NetworkSource:TemplateValue Name="time_type" Type="Type">Time</NetworkSource:Te
mplateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="Coil" UId="29"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="31">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="25" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="32">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="26" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="33">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="27" Name="PT"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="34">
<NetworkSource:NameCon UId="27" Name="ET"/>
<NetworkSource:OpenCon UId="30"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="35">
<NetworkSource:NameCon UId="27" Name="Q"/>
<NetworkSource:NameCon UId="29" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="36">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="29" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="37">
<NetworkSource:NameCon UId="25" Name="out"/>
```

```
<NetworkSource:NameCon UId="26" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="38">
<NetworkSource:NameCon UId="26" Name="out"/>
<NetworkSource:NameCon UId="27" Name="IN"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="39">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="25" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="4" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="5" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="6" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="7" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Error check for feedback On</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="8" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="indErr"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statExternalError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="Coil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
```

<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="9" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="A" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="B" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="C" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>External Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="D" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statExternalError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statFeedbackError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>

```
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="24"/>
<NetworkSource:Part Name="Contact" UId="25"/>
<NetworkSource:Part Name="O" UId="26">
<NetworkSource:TemplateValue Name="Card" Type="Cardinality">2</NetworkSource:Tem
plateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="SCoil" UId="27"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="28">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="29">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="25" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="30">
<NetworkSource:NameCon UId="26" Name="out"/>
<NetworkSource:NameCon UId="27" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="31">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="27" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="32">
<NetworkSource:NameCon UId="24" Name="out"/>
<NetworkSource:NameCon UId="26" Name="in1"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="33">
<NetworkSource:NameCon UId="25" Name="out"/>
<NetworkSource:NameCon UId="26" Name="in2"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="34">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="24" Name="in"/>
<NetworkSource:NameCon UId="25" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="E" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="10" CompositionName="Title">
<ObjectList>
```

```
<MultilingualTextItem ID="11" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="12" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="cmdAut"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutTrig"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Auto"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Man"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="25">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="26"/>
<NetworkSource:Part Name="PBox" UId="27"/>
<NetworkSource:Part Name="Contact" UId="28"/>
<NetworkSource:Part Name="O" UId="29">
<NetworkSource:TemplateValue Name="Card" Type="Cardinality">2</NetworkSource:Tem
plateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="30">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="SCoil" UId="31"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="32">
<NetworkSource:IdentCon UId="21"/>
```

```
<NetworkSource:NameCon UId="26" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="33">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="27" Name="bit"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="34">
<NetworkSource:NameCon UId="27" Name="out"/>
<NetworkSource:NameCon UId="29" Name="in1"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="35">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="28" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="36">
<NetworkSource:NameCon UId="29" Name="out"/>
<NetworkSource:NameCon UId="30" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="37">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="30" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="38">
<NetworkSource:IdentCon UId="25"/>
<NetworkSource:NameCon UId="31" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="39">
<NetworkSource:NameCon UId="26" Name="out"/>
<NetworkSource:NameCon UId="27" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="40">
<NetworkSource:NameCon UId="28" Name="out"/>
<NetworkSource:NameCon UId="29" Name="in2"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="41">
<NetworkSource:NameCon UId="30" Name="out"/>
<NetworkSource:NameCon UId="31" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="42">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="26" Name="in"/>
<NetworkSource:NameCon UId="28" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="13" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="14" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
```

```
<MultilingualText ID="15" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="16" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Auto mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="17" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Man"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="RCoil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="18" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="19" CompositionName="Items">
<AttributeList>
```

```
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="1A" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="1B" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Manual mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="1C" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="ManStart"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="cmdAutOn"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="25">
<NetworkSource:Symbol>
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="26">
<NetworkSource:Symbol>
<NetworkSource:Component Name="indInterlock"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="27">
<NetworkSource:Symbol>
<NetworkSource:Component Name="On"/>
</NetworkSource:Symbol>
```

```
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="28">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="29"/>
<NetworkSource:Part Name="Contact" UId="30"/>
<NetworkSource:Part Name="Contact" UId="31"/>
<NetworkSource:Part Name="O" UId="32">
<NetworkSource:TemplateValue Name="Card" Type="Cardinality">2</NetworkSource:Tem
plateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="33">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="34">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Coil" UId="35"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="36">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="28" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="37">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="29" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="38">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="30" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="39">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="31" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="40">
<NetworkSource:NameCon UId="32" Name="out"/>
<NetworkSource:NameCon UId="33" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="41">
<NetworkSource:IdentCon UId="25"/>
<NetworkSource:NameCon UId="33" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="42">
<NetworkSource:IdentCon UId="26"/>
<NetworkSource:NameCon UId="34" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="43">
<NetworkSource:IdentCon UId="27"/>
<NetworkSource:NameCon UId="35" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="44">
<NetworkSource:NameCon UId="28" Name="out"/>
<NetworkSource:NameCon UId="29" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="45">
<NetworkSource:NameCon UId="29" Name="out"/>
<NetworkSource:NameCon UId="32" Name="in1"/>
```

```
</NetworkSource:Wire>
<NetworkSource:Wire UId="46">
<NetworkSource:NameCon UId="30" Name="out"/>
<NetworkSource:NameCon UId="31" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="47">
<NetworkSource:NameCon UId="31" Name="out"/>
<NetworkSource:NameCon UId="32" Name="in2"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="48">
<NetworkSource:NameCon UId="33" Name="out"/>
<NetworkSource:NameCon UId="34" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="49">
<NetworkSource:NameCon UId="34" Name="out"/>
<NetworkSource:NameCon UId="35" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="50">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="28" Name="in"/>
<NetworkSource:NameCon UId="30" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="1D" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="1E" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="1F" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="20" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Start pump</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="21" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
```

```
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="ManStop"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="cmdAutOn"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="25">
<NetworkSource:Symbol>
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="26">
<NetworkSource:Symbol>
<NetworkSource:Component Name="indInterlock"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="27">
<NetworkSource:Symbol>
<NetworkSource:Component Name="On"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="28">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="29"/>
<NetworkSource:Part Name="Contact" UId="30"/>
<NetworkSource:Part Name="Contact" UId="31">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="O" UId="32">
<NetworkSource:TemplateValue Name="Card" Type="Cardinality">2</NetworkSource:Tem
plateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="33">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="34">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="RCoil" UId="35"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="36">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="28" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="37">
```

```
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="29" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="38">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="30" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="39">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="31" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="40">
<NetworkSource:NameCon UId="32" Name="out"/>
<NetworkSource:NameCon UId="33" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="41">
<NetworkSource:IdentCon UId="25"/>
<NetworkSource:NameCon UId="33" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="42">
<NetworkSource:IdentCon UId="26"/>
<NetworkSource:NameCon UId="34" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="43">
<NetworkSource:IdentCon UId="27"/>
<NetworkSource:NameCon UId="35" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="44">
<NetworkSource:NameCon UId="28" Name="out"/>
<NetworkSource:NameCon UId="29" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="45">
<NetworkSource:NameCon UId="29" Name="out"/>
<NetworkSource:NameCon UId="32" Name="in1"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="46">
<NetworkSource:NameCon UId="30" Name="out"/>
<NetworkSource:NameCon UId="31" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="47">
<NetworkSource:NameCon UId="31" Name="out"/>
<NetworkSource:NameCon UId="32" Name="in2"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="48">
<NetworkSource:NameCon UId="33" Name="out"/>
<NetworkSource:NameCon UId="34" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="49">
<NetworkSource:NameCon UId="34" Name="out"/>
<NetworkSource:NameCon UId="35" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="50">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="28" Name="in"/>
<NetworkSource:NameCon UId="30" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
```

```
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="22" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="23" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="24" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="25" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Stop pump</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="26" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="cmdReset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Reset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="reset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statResetTrig"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="25">
<NetworkSource:Symbol>
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="26"/>
<NetworkSource:Part Name="Contact" UId="27"/>
```

```
<NetworkSource:Part Name="O" UId="28">
<NetworkSource:TemplateValue Name="Card" Type="Cardinality">2</NetworkSource:Tem
plateValue>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="29"/>
<NetworkSource:Part Name="PBox" UId="30"/>
<NetworkSource:Part Name="RCoil" UId="31"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="32">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="26" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="33">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="27" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="34">
<NetworkSource:NameCon UId="28" Name="out"/>
<NetworkSource:NameCon UId="29" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="35">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="29" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="36">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="30" Name="bit"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="37">
<NetworkSource:NameCon UId="30" Name="out"/>
<NetworkSource:NameCon UId="31" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="38">
<NetworkSource:IdentCon UId="25"/>
<NetworkSource:NameCon UId="31" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="39">
<NetworkSource:NameCon UId="26" Name="out"/>
<NetworkSource:NameCon UId="28" Name="in1"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="40">
<NetworkSource:NameCon UId="27" Name="out"/>
<NetworkSource:NameCon UId="28" Name="in2"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="41">
<NetworkSource:NameCon UId="29" Name="out"/>
<NetworkSource:NameCon UId="30" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="42">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="26" Name="in"/>
<NetworkSource:NameCon UId="27" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
```

```
<ObjectList>
<MultilingualText ID="27" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="28" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="29" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="2A" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="2B" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="modeAut"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="Coil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
```

```
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="2C" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2D" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="2E" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="2F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Status mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="30" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statExternalError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="23">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statFeedbackError"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="24">
<NetworkSource:Symbol>
<NetworkSource:Component Name="reset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="25"/>
<NetworkSource:Part Name="Contact" UId="26">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
<NetworkSource:Part Name="Contact" UId="27">
<NetworkSource:Negated Name="operand"/>
</NetworkSource:Part>
```

<NetworkSource:Part Name="Coil" UId="28"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="29">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="25" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="30">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="26" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="31">
<NetworkSource:IdentCon UId="23"/>
<NetworkSource:NameCon UId="27" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="32">
<NetworkSource:IdentCon UId="24"/>
<NetworkSource:NameCon UId="28" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="33">
<NetworkSource:NameCon UId="25" Name="out"/>
<NetworkSource:NameCon UId="26" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="34">
<NetworkSource:NameCon UId="26" Name="out"/>
<NetworkSource:NameCon UId="27" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="35">
<NetworkSource:NameCon UId="27" Name="out"/>
<NetworkSource:NameCon UId="28" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="36">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="25" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="31" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="32" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="33" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="34" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset is needed</Text>
</AttributeList>
</MultilingualTextItem>

```
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="35" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="Err"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="stat"/>
<NetworkSource:Component Name="Error"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="Coil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="36" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="37" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="38" CompositionName="Title">
<ObjectList>
```

```
<MultilingualTextItem ID="39" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI Stat Error</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="3A" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="statAutoMode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="stat"/>
<NetworkSource:Component Name="Automode"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="Coil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="3B" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="3C" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
```

```
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3D" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="3E" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI status mode</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="3F" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="reset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="stat"/>
<NetworkSource:Component Name="Reset"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="Coil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
```

```
<MultilingualText ID="40" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="41" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="42" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="43" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>HMI Reset is needed</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="44" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Auto"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Auto"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="RCoil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
```

```
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="45" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="46" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="47" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="48" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset HMI cmd auto</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<SW.Blocks.CompileUnit ID="49" CompositionName="CompileUnits">
<AttributeList>
<NetworkSource>
<NetworkSource:FlgNet>
<NetworkSource:Parts>
<NetworkSource:Access Scope="LocalVariable" UId="21">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Man"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Access Scope="LocalVariable" UId="22">
<NetworkSource:Symbol>
<NetworkSource:Component Name="HMIData"/>
<NetworkSource:Component Name="cmd"/>
<NetworkSource:Component Name="Man"/>
</NetworkSource:Symbol>
</NetworkSource:Access>
<NetworkSource:Part Name="Contact" UId="23"/>
<NetworkSource:Part Name="RCoil" UId="24"/>
</NetworkSource:Parts>
<NetworkSource:Wires>
<NetworkSource:Wire UId="25">
<NetworkSource:IdentCon UId="21"/>
<NetworkSource:NameCon UId="23" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="26">
<NetworkSource:IdentCon UId="22"/>
```

```
<NetworkSource:NameCon UId="24" Name="operand"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="27">
<NetworkSource:NameCon UId="23" Name="out"/>
<NetworkSource:NameCon UId="24" Name="in"/>
</NetworkSource:Wire>
<NetworkSource:Wire UId="28">
<NetworkSource:Powerrail/>
<NetworkSource:NameCon UId="23" Name="in"/>
</NetworkSource:Wire>
</NetworkSource:Wires>
</NetworkSource:FlgNet>
</NetworkSource>
<ProgrammingLanguage>LAD</ProgrammingLanguage>
</AttributeList>
<ObjectList>
<MultilingualText ID="4A" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="4B" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text/>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="4C" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4D" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>Reset HMI cmd manual</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<MultilingualText ID="4E" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4F" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text>PumpControl</Text>
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.FB>
</Document>
```

Bilag 7: System prompt første udkast

Your role is a programming assistant that can create code that follows the provided rules and syntax in your knowledge base. You will use your extended knowledge to create logic that solves the user function or

problem. You may not create syntax that is not in your knowledge base, but you can be creative of how you connect each instructions/syntax to get the logic working. See it as you have a new way of coding.

Let's think step by step to solve the logic needed and use the syntax provided. Create one network at a time so you are consistent with the syntax but keep in mind the overall code needs to be solved. When you are sure that the code is correct, and the logic is done write """FINAL CODE"""

"""Important"""

Before answering and working on the code, extract and understand the rules and the example in your knowledge base for you to be able to code...

"""

Wait for a response from the user before moving on to the next step.

"""

When you start working on the code do it by these steps one by one:

1. Start by Consulting the Knowledge Base: Before generating any code, refer to the knowledge base for specific syntax rules, instructions, and examples. This ensures compatibility with the code syntax.

2. Create a detailed description for confirmation of the function needed of the user, when the user accepts continue to step 3.

3. Create a self-contained function block that contain the needed in and outputs as well static/temp data and correct data types. When user accept the Block interface continue to step 4 to start creating the networks and logic inside the block.

4. Use Proper Syntax for Instructions: Adhere to the syntax rules for operations, including timers, counters, logic operations, and data manipulation. The instructions within the function block must comply with the examples and formats provided in the knowledge base. you may use your generel knowledge for the logic but always adheres to the rule and instructions for correct syntax and do not use normal STL syntax.

Bilag 8: System prompt sidste udkast

Your role is a programming assistant specializing in PLC (Programmable Logic Controller) software development, specifically for Siemens PLCs. You will create code based on syntax and constraints provided

by the user or your knowledgebase, which ensures compatibility of AWL that can be converted to Ladder Diagram (LAD) formats. Your goal is to generate AWL files based on the steps that you need to follow step by step, always ensure to use the user defined rules and syntax when working on the logic and not use common STL code.

Here is how you should approach the task:

1. **Review User-Provided Examples**:

Before beginning any coding, review the syntax rules, instructions, and examples provided by the user or in your knowledge base to ensure that your code will be compatible with the necessary PLC programming formats.

2. **Define the Function Block Interface**:

Work with the user to define and confirm the necessary inputs, outputs, and functionalities of the function block. before moving on to the next step, wait for the user to verify the function and interface of the block. GPT can even sugets more functions to the block using the domain knowledge of PLC and industrial standards

3. **Develop the Function Block**:

Construct a self-contained function block that incorporates all required inputs, outputs, internal variables, and logic. Always utilize the correct data types and adhere to the syntax that has been provided by the user or your knowledgebase. Also use the Programming_Styleguide from Siemens in your knowledgebase for good practice for siemens PLC programmering, such as camelCasing for variables and PascaleCasing for blocks, change log in the blocks description and so on, focus on LAD. Remember to add comment to variables and networks

4. **Implement Logic Using Appropriate Syntax**:

Carefully apply the syntax rules for operations, including timers, counters and logic operations. Ensure that every network element within the function block complies with the formats provided by the user, using general programming knowledge to shape the logic but adhering strictly to the specifics of PLC programming syntax provided by the user or in your knowledgebase "Operations and instructions".

5. ** Check the code for common Errors made by the GPT**

Common errors when generating the code:

- Not paying attention to the knowledge base or user provided syntax.

- When calling a function such as a timer (TON) or counter (CTU) see the knowledgebase for correct use, often mistakes is missing the comma for separating the parameters, forgetting the use of local memory %L0.x when a bool input is needed and BLD 103 after each bool logic. When finished calling the function remember the NOP 0 instruction

- Using logic that is SCL or STL that cannot be converted to LAD, therefore use the provided example to check how to build the logic

- Generel issues when generating logic from AWL to LAD is missing 2 key point. multiple assignment in one network and not using bracket for OR operations and so on, here is some more information to guide the GPT:

The below examples explain some of the reasons why an STL program cannot be displayed in LAD. Although you can set the view to "LAD" in the LAD/STL/FBD editor, the program code appears completely or in "STL" only in some networks.

No 1

Multiple assignments programmed in one network

A STEP 7 program written in STL is supposed to be displayed in LAD. However, after STL programming it is not possible to switch to LAD.

One reason for this might be that a new instruction was begun with after an "S" or "R" assignment. in LAD, a new network is begun with after each "S", "R" or "=" assignment, because only one of these assignments is permitted in one network. In STL you can program the program code with any length and with multiple assignments. The example shows an STL program in which after the assignment "S #Output1" the next subprogram (instruction "A #input3") begins. It is then no longer possible to switch from STL to LAD:

```
NETWORK
   TITLE = Incorrect use of assignment
      // comment
      A     #input1;
      AN     #input2;
      S     #Output1;
      A     #input3;
      A     #input1;
      =     #output2;
```

Fig. 01

Remedy

Divide your STL program into the relevant networks so that a new network begins after each assignment ("S", "R" or "="). If the program code is in a second network from instruction "A #input1" onwards as in the example, it is possible to switch from STL LAD.

```
NETWORK
   TITLE = Correct use of assignment
      // comment
```

```
    A    #input1;
    AN    #input2;
    S    #Output1;
NETWORK
  TITLE = Correct use of assignment
    // comment
    A    #input3;
    A    #input1;
    =    #output2;
```

No 2

Another reason might be the use of instructions in a sequence that is not sufficiently structured. You can configure the program code more freely in STL than in other programming languages.

```
NETWORK
  TITLE = Incorrect use of logic
    // comment
    A    #input1;
    A    #input2;
    O    #input3;
    A    #input4;
    =    #Output1;
```

Remedy

Use a structured sequence and brackets when programming multiple instructions.

```
NETWORK
  TITLE = Correct use of logic
    // comment
    A(
    A    #input1;
    A    #input2;
    O    #input3;
```

```
)
A    #input4;
=    #Output1;
```

6. **Review generated code to adhere the knowledge base**:

Invoke a different view on the code and be critical. See it as a co-worker reviewing the code the GPT just created. the re-viewer have deep understanding of the rules and syntaxes that can and cannot be used for this code to able to convert to LAD

6. **Iterative Review and Confirmation**:

Engage in a continuous feedback loop with the user to review the proposed logic, making adjustments as necessary to ensure that the function block meets all operational requirements and is free of syntactic errors.

7. **implementation the code**:

When done with the code, use your code tool to create a new text file with the file extension .awl and past the code inside of this file. Ask the user if it is the first time implementing a source file to TIA portal, if so, guide the user as follows:

1. Open TIA portal

2. Add a S7 300 or 400 serie PLC

3. Open the "External source file" and click add

4. Browse to the file that you saved from this chat or copy the code from the to a text file and change the file extension to .awl.

5. Right click on the file and chose "Generate blocks from source"

6. Right click on the new block and click on "Switch programming language" to LAD