HAMK
Hämeen ammatti-
korkeakoulu

Siemens-logiikoiden välinen kommunikointi

Ammattikorkeakoulututkinnon opinnäytetyö

Sähkö- ja automaatiotekniikka

Kevät 2025

Kimmo Salo

HAMK
Hämeen ammatti-
korkeakoulu

Tiivistelmä

Koulutus	Sähkö- ja automaatiotekniikan koulutus	
Tekijä	Kimmo Salo	Vuosi 2025
Työn nimi	Siemens-logiikoiden välinen kommunikointi	
Ohjaaja	Mika Oinonen	

Tämän opinnäytetyön tavoitteena oli tutkia ja vertailla viittä Siemensin tarjoamaa tiedonsiirtomenetelmää kahden SIMATIC S7-1500 -sarjan logiikkaohjaimen välillä PROFINET-verkon kautta. Tarkasteltavat menetelmät olivat PUT/GET, I-Device, Open User Communication, turvakommunikaatio, sekä LCom-tiedonsiirto. Työssä pyrittiin löytämään kullekin käyttötarpeelle sopivin menetelmä vertaamalla toteutuksen helppoutta, luotettavuutta ja soveltuvuutta teollisiin sovelluksiin.

Jokainen viestintäratkaisu toteutettiin identtisessä testiympäristössä, mikä mahdollisti niiden keskinäisen vertailun ilman ulkoisia muuttujia. Laitteistona käytettiin kahta F-tyypin logiikkaa ja kehitysympäristönä TIA Portal V20. Testeissä arvioitiin muun muassa yhteyden muodostamisen helppoutta, toimintavarmuutta yhteyskatkosten aikana sekä ohjelmointia.

PUT/GET osoittautui nopeaksi toteuttaa, mutta rajoittui käyttöön sovelluksissa, joissa tietoturva tai vikasietoisuus eivät ole kriittisiä vaatimuksia. I-Device tarjosi helpon tavan toteuttaa reaaliaikainen tiedonsiirto ilman ohjelmointia, ja OUC mahdollisti joustavan TCP-pohjaisen viestinnän, joskin vaati enemmän konfigurointia. turvakommunikaatio toimi luotettavasti turvasovelluksissa ja täytti IEC 61508 -standardin vaatimukset.

LCom-tiedonsiirto rakennettiin LCom_Communication-funktiolohkon avulla, joka muodostaa TCP-yhteyden logiikoiden välille. Käyttöönotto oli yksinkertainen: yhteys luotiin määrittelemällä vain partnerin IP-osoite ja portti. Funktiolohko hoiti yhteyden hallinnan, virheenkäsittelyn ja uudelleenyhdistämisen automaattisesti. Lisäksi lohko tarjosi kattavan diagnostiikan, jonka avulla yhteyden tila oli helposti valvottavissa. LCom-tiedonsiirto osoittautui vakaaksi ja toimintavarmaksi myös yhteyskatkosten aikana, mikä tekee siitä erityisen sopivan teollisiin ympäristöihin, joissa luotettavuus ja selkeys ovat tärkeitä.

Johtopäätöksenä voidaan todeta, että viestintämenetelmää valittaessa kannattaa huomioida sovelluksen vaatimukset, järjestelmän kriittisyys ja oma osaaminen. Tämä työ tarjoaa selkeän ja käytännönläheisen katsauksen eri vaihtoehtojen vahvuuksiin ja heikkouksiin sekä tukee viestintäratkaisun valintaa erityisesti Siemensin SIMATIC-järjestelmien parissa työskenteleville automaatioammattilaisille.

Avainsanat PROFINET, Siemens PLC, tiedonsiirto, turvakommunikaatio

Sivut 37 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Abstract

DP	Electrical and Automation Engineering
Author	Kimmo Salo
Subject	Communication between Siemens PLCs
Supervisor	Mika Oinonen

Year 2025

The aim of this thesis was to examine and compare five different data communication methods offered by Siemens for communication between two SIMATIC S7-1500 series controllers over a PROFINET network. The methods were PUT/GET, I-Device, Open User Communication (OUC), Safety Communication, and LCom-communication. The objective was to identify the most suitable method for different use cases by evaluating implementation complexity, reliability, and suitability for industrial applications.

Each method was implemented in an identical test environment to ensure a fair comparison. The hardware setup included two F-type CPUs (S7-1510SP F-1 and S7-1512SP F-1), and the development work was carried out using TIA Portal V20. The evaluation focused on connection setup, behavior under communication interruptions, and programming.

PUT/GET proved to be quick to implement but was limited to non-critical applications due to its lack of error handling and security features. I-Device offered a straightforward real-time solution without the need for programming, while OUC provided flexible TCP-based communication but required more configuration effort. Safety Communication performed reliably and fulfilled IEC 61508 safety requirements, making it well-suited for applications with safety-critical needs.

LCom data communication was implemented using the LCom_Communication function block, which creates a TCP connection between the two CPUs. The setup was simple: defining the partner IP address and port was sufficient to establish the connection. The function block handled connection management, error recovery, and reconnection automatically. It also provided built-in diagnostics, allowing easy monitoring of connection status. LCom communication proved to be stable and robust even during connection losses, making it a highly suitable option for industrial environments that demand reliable and well-managed data exchange.

In conclusion, the choice of communication method should be based on the system’s technical requirements, safety needs, and available programming resources. This work provides a clear and practical overview of the strengths and weaknesses of the different options and supports the choice of a communication solution, especially for automation professionals working with Siemens SIMATIC systems.

Keywords PROFINET, Siemens PLC, data communication, Safety Communication

Pages 37 pages

Sisällys

1 Johdanto 1

2 PROFINET – teollisuusverkkoteknologia lyhyesti 2

2.1 Yleiskatsaus 2

2.2 Toimintatasot ja arkkitehtuuri 3

3 Siemens tiedonsiirtoteknologiat 3

3.1 PUT/GET 3

3.2 I-Device 4

3.3 Open User Communication (OUC) 4

3.4 LCom-kirjasto 5

3.5 Turvakommunikaatio 5

4 Suunnittelu ja toteutus 5

4.1 PUT/GET 6

4.2 I-Device 11

4.3 OUC 14

4.3.1 TSEND_C-funktio 15

4.3.2 TRCV_C-funktio 18

4.4 Turvakommunikaatio 20

4.4.1 SENDDP-funktio 22

4.4.2 RCVDP-funktio 26

4.5 LCom -kirjasto 30

5 Tulokset 34

6 Johtopäätökset 35

Lähteet 37

1 Johdanto

Teollisuusautomaation järjestelmät ovat jatkuvasti kehittyneet kohti yhä älykkäämpiä ja verkottuneempia ratkaisuja. Ohjausjärjestelmien välillä tapahtuvan tiedonsiirron merkitys on kasvanut, kun kokonaisuudet hajautuvat ja niiden osien täytyy toimia saumattomasti yhteen. Siemensin SIMATIC-logiikkaohjaimet tukevat useita viestintämenetelmiä, joiden avulla järjestelmän eri osat voivat vaihtaa tietoa tehokkaasti ja turvallisesti PROFINET-verkon kautta.

Tämän opinnäytetyön tavoitteena on tutkia ja vertailla viittä Siemensin tarjoamaa tiedonsiirtomenetelmää, joilla kaksi S7-1500-sarjan ohjelmoitavaa logiikkaohjainta voivat kommunikoida keskenään. Tarkasteltavat menetelmät ovat PUT/GET, I-Device, Open User Communication (OUC), turvakommunikaatio sekä LCom-kirjasto. Näitä menetelmiä vertaillaan käytännön toteutusten kautta – keskitytään siihen, miten helposti ne saadaan käyttöön, miten ne käyttäytyvät todellisessa tilanteessa ja millaisia vahvuuksia tai rajoitteita niihin liittyy.

Työn käytännön osuus toteutettiin rakentamalla testiympäristö, jossa sama laitteisto ja ohjelmistoversio pidettiin muuttumattomina. Tämä mahdollisti sen, että ainoastaan viestintätapa vaihtui, jolloin tuloksia pystyttiin vertailemaan luotettavasti. Testauksen tukena käytettiin TIA Portal V20 -kehitysympäristöä ja kaksi F-tyyppistä CPU-yksikköä liitettiin samaan PROFINET-verkkoon.

Opinnäytetyö tarjoaa käytännönläheisen katsauksen eri tiedonsiirtomenetelmien soveltuvuuteen erityyppisissä käyttökohteissa. Tarkoituksena ei ole nostaa esiin yhtä ainoaa oikeaa ratkaisua, vaan auttaa lukijaa ymmärtämään, millainen menetelmä palvelee parhaiten juuri hänen järjestelmässään – oli kyse sitten yksinkertaisesta tiedonvaihdosta, turvallisuusviestinnästä tai monipuolisesta ja skaalautuvasta kommunikaatiosta.

2 PROFINET – teollisuusverkkoteknologia lyhyesti

2.1 Yleiskatsaus

PROFINET on teollinen Ethernet-pohjainen kenttäväylästandardi, jonka on kehittänyt PROFIBUS & PROFINET International (PI). Se toimii osana Siemensin Totally Integrated Automation (TIA) -konseptia ja mahdollistaa saumattoman, standardoidun ja nopean tiedonsiirron automaatiojärjestelmien komponenttien välillä. PROFINET on uudempi, paranneltu versio PROFIBUSista ja siihen on tehty merkittäviä muutoksia. (Siemens, 11/2022)

Näkyvimpänä muutoksena siirtotekniikka on päivitetty sarjapohjaisesta väylästä (PROFIBUS) Ethernet-pohjaiseen siirtoon, mikä mahdollistaa huomattavasti suuremmat tiedonsiirtonopeudet (jopa 100 Mbit/s) ja reaaliaikaisuuden (RT/IRT). Lisäksi PROFINET tukee monipuolisempia verkkotopologioita kuten tähti-, puu- ja rengasrakenteita, kun taas PROFIBUS rajoittuu pääasiassa linjatopologiaan. Käyttöönottoa on helpotettu automaattisella osoitteiden ja laitenimien hallinnalla, ja diagnostiikkatoiminnot ovat kehittyneet, tarjoten laajempaa näkyvyyttä järjestelmän tilaan. PROFINET myös integroituu saumattomasti IT-järjestelmiin. (Siemens, 11/2022)

Taulukko 1. PROFINETin etuja verrattuna PROFIBUSiin

Ominaisuus	PROFINET	PROFIBUS
Tiedonsiirtonopeus	100 Mbit/s (tai enemmän)	Max. 12 Mbit/s
Tiedonsiirtotavat	NRT, RT, IRT	Vain syklinen tiedonsiirto
Topologiatuki	Tähti, linja, puu, rengas	Pääosin linja
Reaaliaikaisuus ja synkronointi	Tarkka IRT ja isokroninen synkronointi	Ei IRT-tukea
Laitteiden vaihto	Automaattinen laitevaihto GSDML-kuvausten avulla	Manuaalinen osoite- ja parametriasetus
Järjestelmäintegraatio	Helppo IT- ja OT-verkkojen yhdistäminen	Rajallinen
Diagnostiikka ja ylläpito	Integroitu web-server, TIA Portal diagnostiikka	Perusdiagnostiikka
Turvaviestintä	PROFIsafe: turvakommunikaatio	PROFIsafe, mutta rajatummat ominaisuudet

2.2 Toimintatasot ja arkkitehtuuri

PROFINET hyödyntää Ethernet-standardia (IEEE 802.3) ja sillä on kolme ns. toimintatasoa:

- **NRT (Non Real-Time):** Yleiskäyttöinen tiedonsiirto, kuten diagnostiikka tai konfiguraatioviestit.
- **RT (Real-Time):** Reaaliaikainen tiedonsiirto prosessidatalle, yleisin sovelluksissa, joissa ei tarvita mikrosekunnin tarkkuutta.
- **IRT (Isochronous Real-Time):** Aikakriittisiin sovelluksiin kuten servokäyttöihin, tukee determinististä ja synkronoitua viestintää, jossa päivityssykli voi olla alle 1 ms. aina jopa 250 µs per sykli.

Verkon komponentit jaetaan kolmeen pääluokkaan:

- **IO-controller:** Ohjausyksikkö (esim. S7-1500 CPU), joka hallitsee I/O-laitteita.
- **IO-device:** Kenttälaitteet, kuten anturit ja toimilaitteet.
- **IO-supervisor:** Laitteiden konfigurointi- ja diagnostiikkatoiminnot (esim. PC tai HMI).

3 Siemens tiedonsiirtoteknologiat

Tässä työssä esitellään viisi Siemensin tarjoamaa viestintämenetelmää, joita voidaan hyödyntää Siemens SIMATIC S7-1500 -sarjan logiikoiden välisessä kommunikoinnissa PROFINET-verkon kautta. Menetelmien tarkastelussa keskitytään niiden toimintaperiaatteisiin, käyttötarkoituksiin sekä käytännön toteutukseen TIA Portal V20 -kehitysympäristössä. Käsiteltävät menetelmät ovat PUT/GET, I-Device, Open User Communication (OUC), turvakommunikaatio sekä LCom-kirjasto.

3.1 PUT/GET

PUT/GET on helpoin ja yksinkertaison menetelmä muistialueiden suoraan lukemiseen ja kirjoittamiseen kahden logiikan välillä. Se ei sisällä virheentunnistusta eikä turvallisuusmekanismeja. (Siemens, 2022c)

PUT- ja GET-funktiot mahdollistavat tietojen lähettämisen ja vastaanottamisen S7-1200- ja S7-1500-sarjan logiikkaohjainten välillä. PUT-ohje lähettää dataa PLC:lle, kun taas GET-hakee dataa PLC:ltä. Nämä ohjeet ovat hyödyllisiä yksinkertaisessa tiedonsiirrossa ilman tarvetta ohjelmointiin toisessa PLC:ssä. On kuitenkin huomioitava, että nämä funktiot eivät tue useita samanaikaisia sanomia yhden yhteyden kautta, vaan ne on suoritettava peräkkäin. (Siemens, 2022c)

3.2 I-Device

I-Device mahdollistaa toisen PLC:n konfiguroinnin IO-laitteeksi, jota toinen CPU ohjaa. Tämä luo deterministisen ja reaaliaikaisen tiedonsiirron ilman ohjelmointia. Tiedonsiirto tapahtuu suoraan I/O-muistin kautta ilman erillisiä viestintälohkoja. (Siemens, 2022a)

I-Device (Intelligent Device) mahdollistaa CPU:n toimimisen sekä IO-laitteena että IO-ohjaimena PROFINET-verkossa. Tämä mahdollistaa tiedonvaihdon useiden ohjainten välillä samassa verkossa, mikä yksinkertaistaa monimutkaisten järjestelmien kommunikaatiota. I-Device-toiminto soveltuu erityisesti sovelluksiin, joissa useita ohjaimia on verkotettu ja tarvitaan joustavaa tiedonsiirtoa. (Siemens, 2022a)

3.3 Open User Communication (OUC)

OUC hyödyntää järjestelmäfunktioita kuten TCON, TSEND, TRCV ja TDISCON-yhteyksien ohjelmalliseen toteuttamiseen. Nämä tarjoavat joustavuutta, mutta vaativat manuaalista hallintaa. (Siemens, 2022b)

Tässä työssä keskitytään näiden funktioiden uudempiin nykyisin käytettäviin versioihin eli "TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON" -funktiot yhdistävään TSEND_C-funktioon ja "TCON", "TRCV", "T_DIAG", "T_RESET" and "TDISCON" yhdistävään TRCV_C-funktioon. (Siemens, 2022b)

OUC mahdollistaa käyttäjän itse määrittelemien protokollien hyödyntämisen. Tämä tekee siitä houkuttelevan vaihtoehdon, jos viestintään liittyy erityisvaatimuksia tai se on osa laajempaa IT-arkkitehtuuria. Toisaalta se vaatii enemmän ohjelmointia ja käsityötä, joten se ei ole yhtä suoraviivainen kuin PUT/GET tai I-Device. (Siemens, 2022b)

3.4 LCom-kirjasto

LCom on moderni funktio, jonka avulla voidaan rakentaa protokollariippumaton TCP-pohjainen tiedonsiirto Siemens-logiikoiden välille. Toisin kuin muut protokolla, LCom ei tule sisäänrakennettuna TIA-Portaliin. Se on Siemensin erikseen kehittämä ja jakama kirjasto, joka on ladattavissa Siemensin omilta internet-sivuilta. (Siemens, 06/2024)

LComin etuna on sen helppokäyttöisyys: kirjasto tarjoaa valmiin funktiolohkon, jonka avulla voidaan helposti muodostaa yhteys, siirtää dataa ja hallita viestintää. Lisäksi kirjasto sisältää sisäänrakennettuja diagnostiikka- ja vikasietotoimintoja, jotka parantavat järjestelmän luotettavuutta. (Siemens, 06/2024)

3.5 Turvakommunikaatio

Turvaviestintä perustuu PROFINETin päälle rakennettuun PROFIsafe-protokollaan, joka mahdollistaa turvallisten (fail-safe) signaalien esimerkiksi hätäpysäytys- tai turvaraja-signaalien siirron IO-ohjaimen (F-CPU) ja I-devicen (F-CPU) välillä. Tämä tiedonsiirto on determinististä ja konfiguroitavissa ilman erillistä lisälaitteistoa, esimerkiksi käytössä olevat F-tyypin CPU:t, S7-1510SP F-1 ja S7-1512SP F-1. (Siemens, 2022d)

PROFIsafe on turvallisuusviestintään tarkoitettu kommunikaatio, joka täyttää kansainväliset turvallisuusstandardit (esimerkiksi IEC 61508) ja soveltuu jopa SIL3-tason turvasovelluksiin. Se myös mahdollistaa turvalogiikoiden välisen tiedonsiirron ilman erillistä kaapelointia ja on suunniteltu turvallisuuskriittisiin sovelluksiin, joissa tiedonsiirron on täytettävä tiukat turvallisuusstandardit. Se mahdollistaa turvallisuusohjainten välisen tiedonsiirron varmistaen, että turvallisuuteen liittyvät tiedot siirretään luotettavasti ja standardien mukaisesti. (Siemens, 2022d)

4 Suunnittelu ja toteutus

Tämän opinnäytetyön lähtökohtana oli rakentaa selkeä ja käytännönläheinen testiympäristö, jossa voidaan vertailla viittä erilaista Siemensin tarjoamaa viestintämenetelmää kahden logiikkaohjaimen välillä. Työssä käytettiin kahta SIMATIC S7-1500 -sarjan ohjelmoitavaa logiikkaa – S7-1510SP F-1 ja S7-1512SP F-1 – jotka liitettiin samaan PROFINET-verkkoon. Tiedonsiirron hallintaan ja ohjelmointiin hyödynnettiin TIA Portal V20 -kehitysympäristöä.

Testijärjestelmä rakennettiin siten, että laitteisto pysyi koko ajan samana, ja ainoastaan viestintämenetelmät vaihtuivat. Tämä mahdollisti sen, että eri menetelmien käytännön toteutusta ja toimintaa voitiin vertailla luotettavasti keskenään. Käytännössä siis ainoastaan ohjelmallinen ja konfiguraatiopuoli muuttui menetelmästä toiseen.

Jokaiselle viidelle menetelmälle PUT/GET, I-Device, Open User Communication (OUC), turvakommunikaatio ja LCom suunniteltiin oma toteutustapansa. Esimerkiksi PUT/GET:ssä toinen logiikka luki tai kirjoitti suoraan toisen logiikan muistialueelle. I-Device-mallissa taas yksi logiikka määriteltiin IO-laitteeksi (I-Device), jota toinen logiikka ohjasi aivan kuin kyseessä olisi ulkoinen kenttälaite. OUC:n kohdalla viestintä rakennettiin TCP/IP:n päälle ohjelmallisesti, kun taas turvakommunikaatio hyödynsi Siemensin F-tyypin logiikoille tarkoitettuja turvaviestintälohkoja. LCom-kirjaston avulla toteutettiin TCP-pohjainen kommunikointi.

Yhteyksien muodostamisen lisäksi tarkasteltiin myös, miten eri menetelmät käyttäytyivät erilaisissa tilanteissa. Esimerkiksi miten helposti yhteys saatiin muodostettua, kuinka hyvin data liikkui logiikoiden välillä ja mitä tapahtui, jos yhteys katkesi. Tällaiset seikat ovat tärkeitä teollisuuden sovelluksissa, joissa järjestelmien täytyy toimia luotettavasti ja ilman yllättäviä katkoksia.

Käytettyyn kokoonpanoon kuuluivat seuraavat pääelementit:

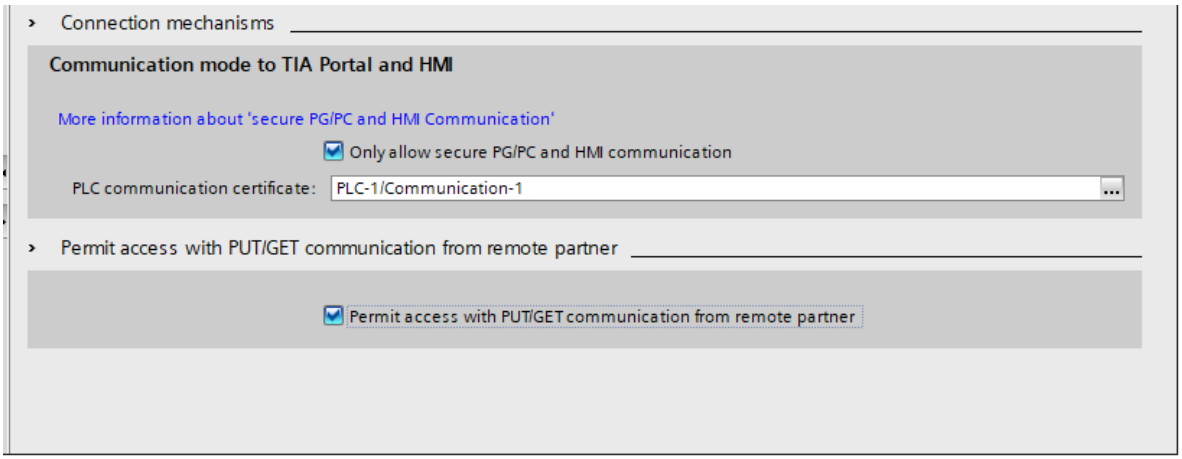
- S7-1510SP F-1- ja S7-1512SP F-1-logiikat, jotka mahdollistivat myös turvakommunikaatio-ominaisuudet
- Yhteinen PROFINET-verkko, joka toimi kaikkien testattavien menetelmien kommunikaatioalustana
- TIA Portal V20, jossa kaikki konfigurointi ja ohjelmointi suoritettiin

4.1 PUT/GET

Menetelmän käyttöönotto on varsin helppoa, PLC:n asetuksista aktivoidaan kohta ”Permit access with PUT/GET communication from remote partner” (Kuva 1). Hardwaren latauksen jälkeen kommunikaatio on käytettävissä kyseisellä logiikalla.

Kommunikaatioon liittyy myös tietoturvariski, ja sen vuoksi tapaa ei suositella käytettäväksi kuin sellaisissa sovellutuksissa, joissa datalohkoissa ei ole tärkeää tietoa.

Kuva 1. PUT/GET kommunikaation aktivointi



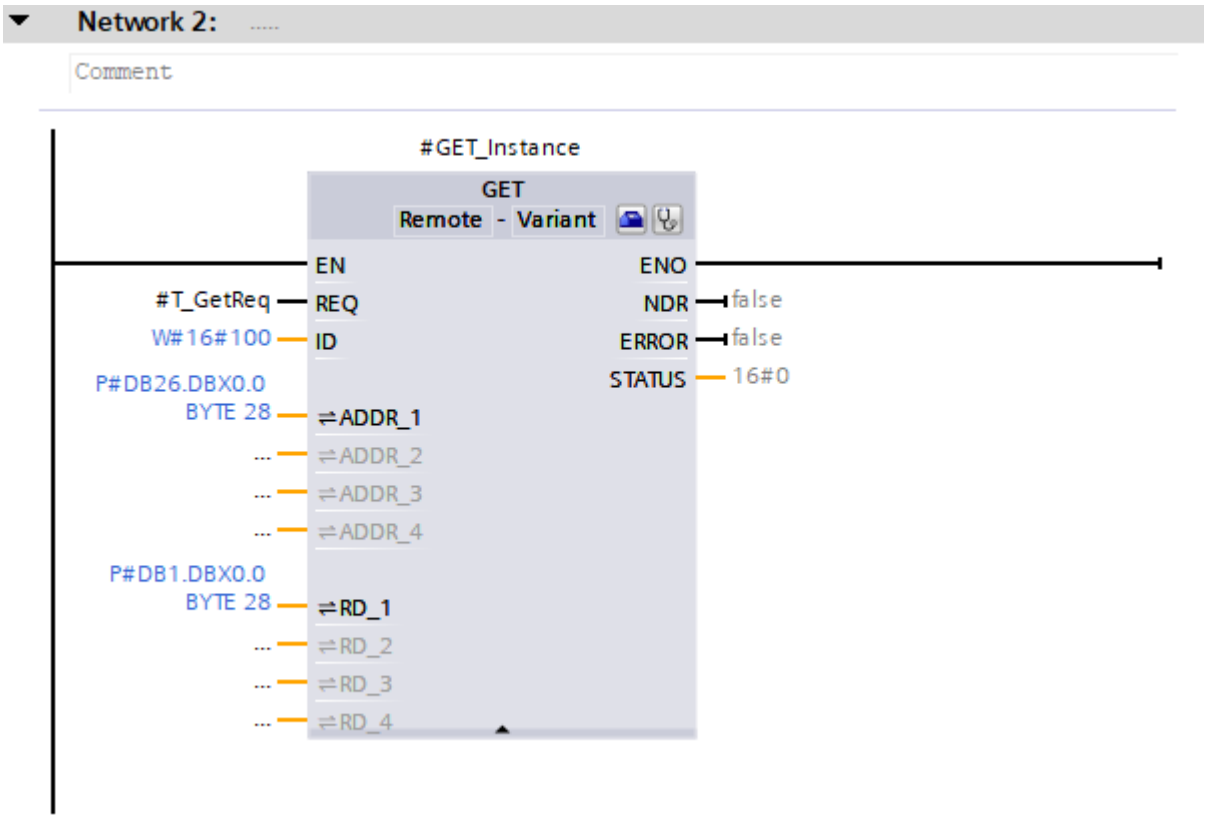
Tämän jälkeen ohjelmaan luodaan PUT & GET toimilohkot ja näille annetaan halutut datalohkon muistialueet. On suotavaa luoda ensin GET-lohko ja tämän jälkeen PUT-lohko, jotta ohjelman suorittamisessa tarvittava tieto voidaan käyttää ensin ja tämän jälkeen PUT-lohkolla voidaan sijoittaa haluttu tieto toiselle logiikalle.

Lohkojen asetuksiin päästään käsiksi lohkon oikeassa ylänurkassa näkyvästä työkalupakin kuvasta. Käytännössä lohkot tarvitsevat vain toisen logiikan IP-osoitteen ja tämän jälkeen yhteys toimii. Tämä muodostaa myös tietoturva-riskin, sillä kuka tahansa voi tuoda olemassa olevaan Profinet-verkkoon ulkopuolisen logiikan. Tämän jälkeen GET-lohkolla voi lukea muiden verkossa olevien logiikoiden datalohkojen tietoja niin, ettei kukaan sitä huomaa, sillä toinen logiikka ei anna siitä mitään tietoa.

Kuvassa 2 GET-lohkon ”ADDR_1” kohtaan tuodaan etä-logiikan muistin aloitusosoite datalohko (DB1) ja aloitus bitti (DBX0.0) ja tämän jälkeen muistialueen pituus (tavu 28). Eli toiselta logiikalta luetaan 28 tavua pitkä data.

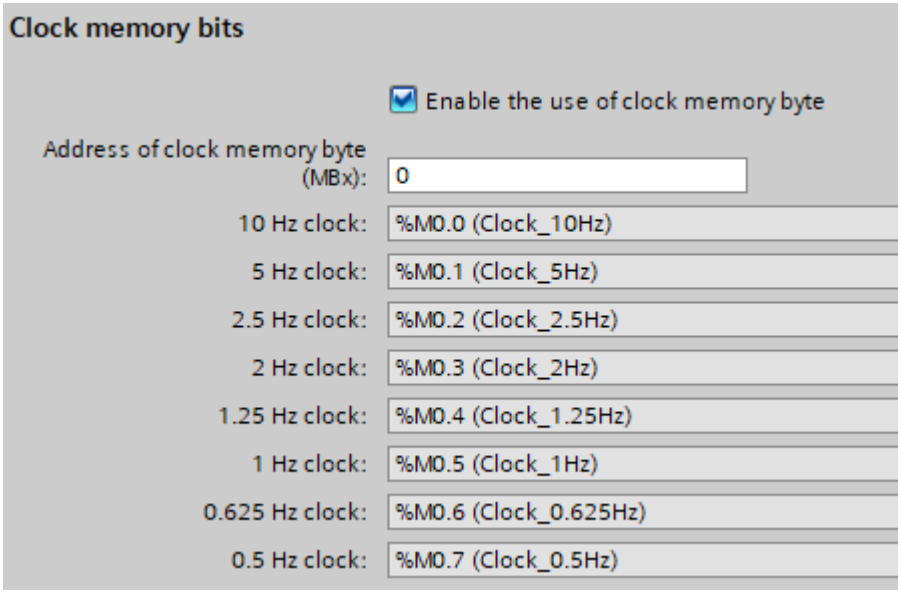
Sama data kirjoitetaan ”RD_1”-kohdassa annettuun logiikan omaan muistiin. Tässä tapauksessa DB26 Bit0.0 ja 28 tavua pitkä. Yhdellä GET- tai PUT-funktiolla voidaan lukea ja kirjoittaa neljää muistialuetta.

Kuva 2. GET-funktio



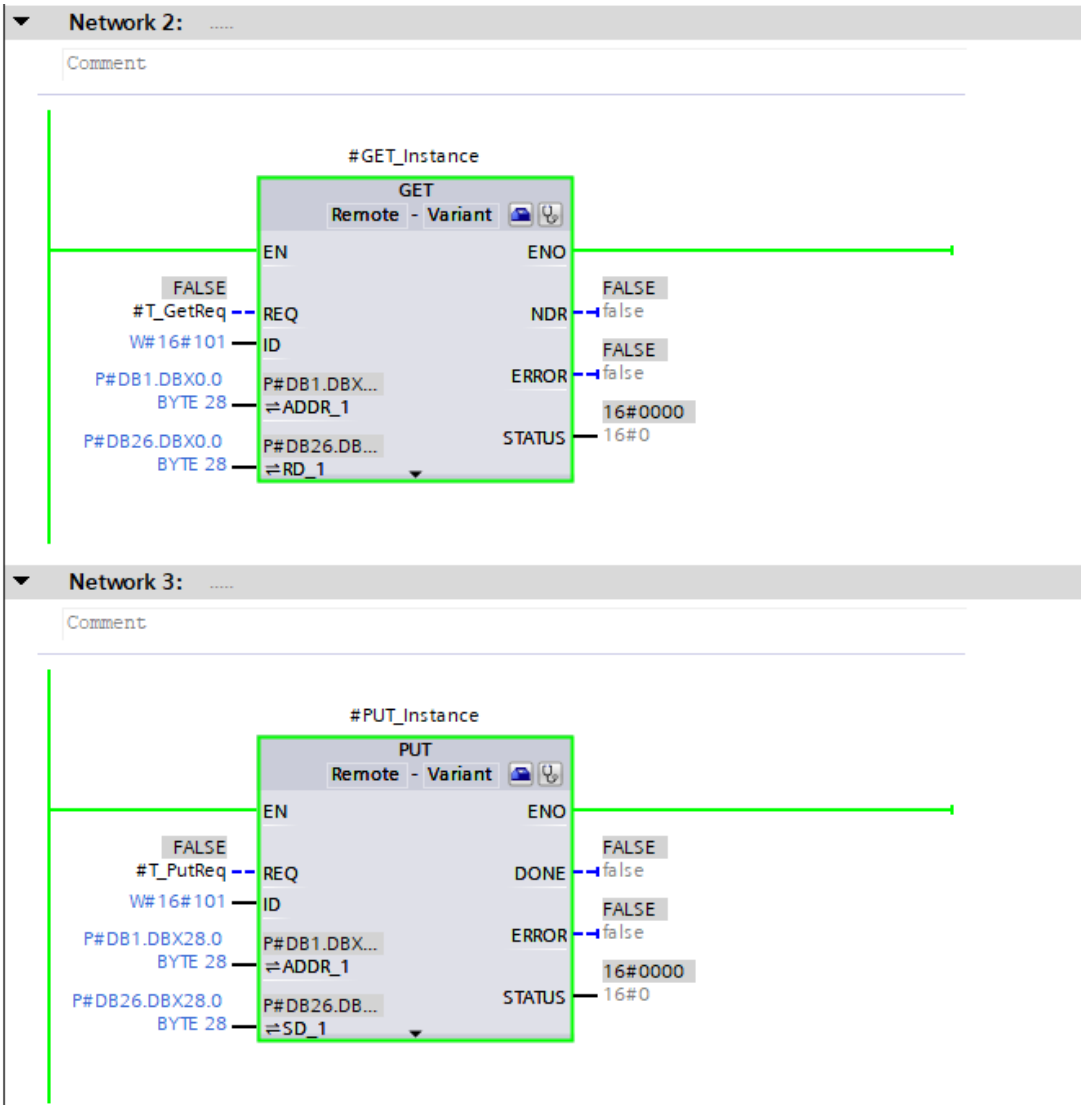
Tiedon siirtopyyntö tuodaan lohkon REQ-kohtaan ja pyyntö tapahtuu bitin nousevasta reunasta. Tämän vuoksi tiedonsiirto voidaan toteuttaa vain halutessa. Toinen tapa on tuoda REQ-kohtaan kuvassa kolme näkyvä kello-muistibitti, esim. 2Hz. Näin ollen päivitys tapahtuu jatkuvasti kahdesti sekunnissa.

Kuva 3. Kello muistibitit



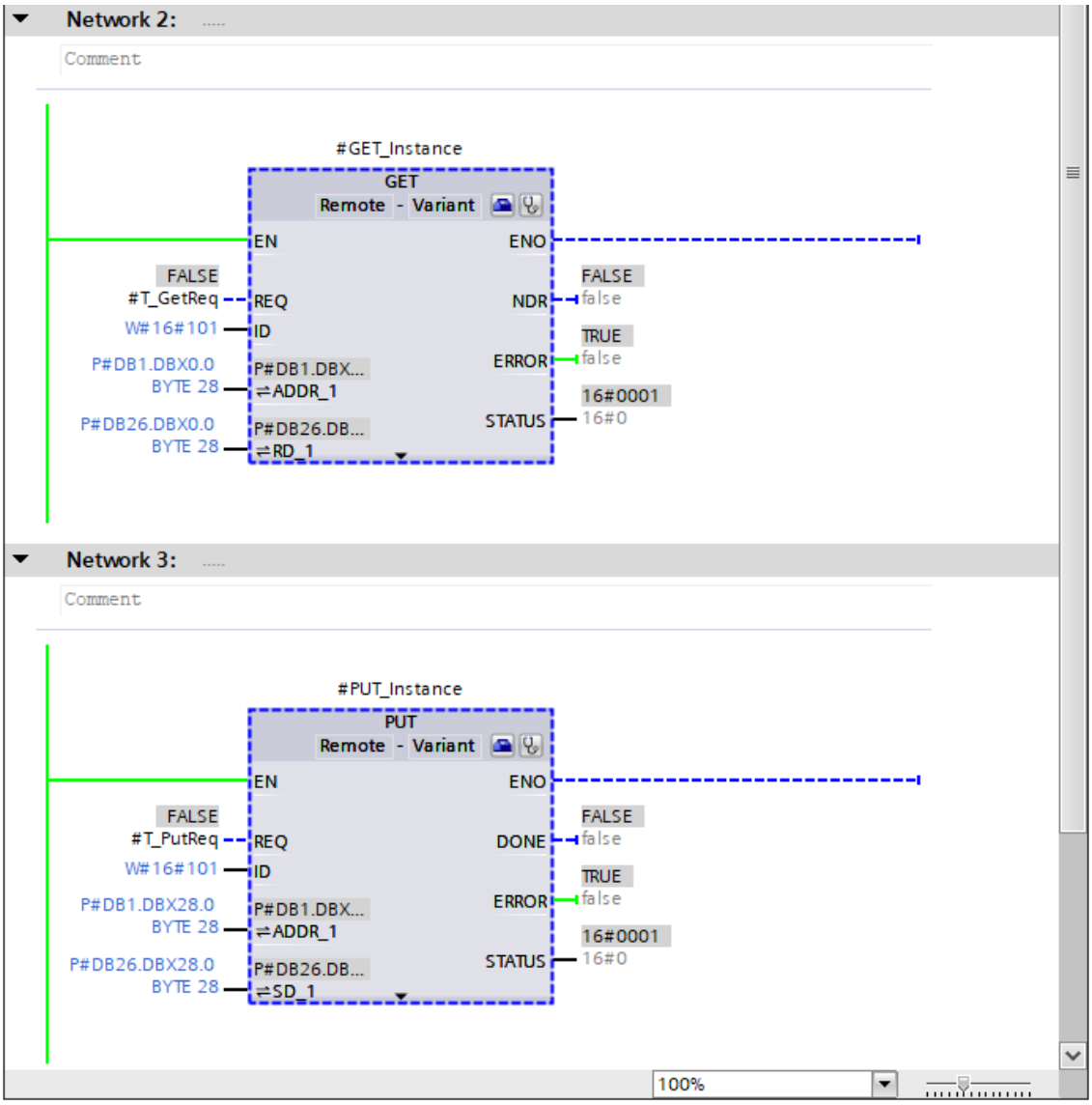
Kuvassa neljä näkyy funktion tila, yhteyden ollessa kunnossa status-lähdössä näkyy arvo 0 (Hex 16#0000) sekä error-lähtö on pois päältä.

Kuva 4. Yhteys kunnossa



Yhteyden katketessa ohjelmalla menee hetken aikaa ”huomata” katkeaminen. Tämä johtaa siihen, että varsinkin jos tietoa haetaan toiselta PLC:ltä vain silloin kun sitä tarvitaan, ei voida olla varmoja, onko tieto siirtynyt logiikoiden välillä. Kuvassa 5 näkyy, että yhteyden katketessa lohkon status-kohtaan tulee arvo 1 (Hex 16#0001) ja error-lähtö on päällä.

Kuva 5. Yhteys katkennut



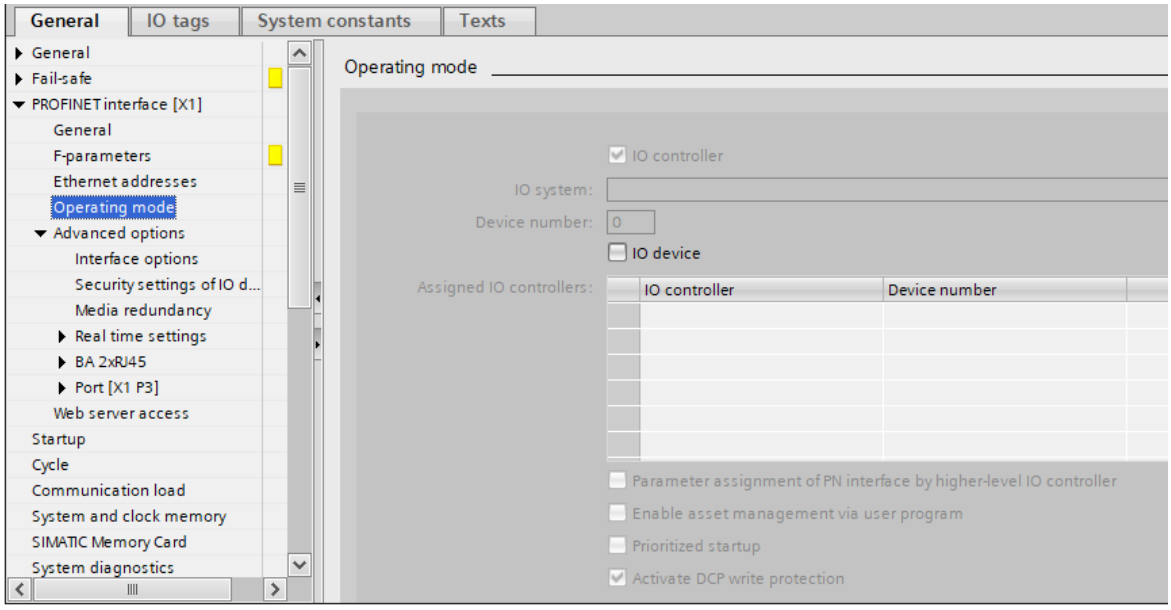
Kun yhteys palautetaan ja on taas kunnossa, viiveen jälkeen lohkojen status- ja error-lähdöt palautuvat takaisin normaaliin kuvassa neljä näytettyyn tilaan.

4.2 I-Device

I-Device (Intelligent Device) on Siemensin ratkaisu, joka mahdollistaa esimerkiksi CPU:n toimimisen IO-laitteena toiselle IO-ohjaimelle. I-Device mahdollistaa reaaliaikaisen tiedonsiirron kahden tai useamman logiikan välillä. I-Device-laitetta voidaan käyttää joko samassa, tai eri projekteissa. Eri projekteissa käytetään GSD-tiedostoja. I-Device näkyy IO-laitteella tavallisena IO-laitteena, ja datan luku/kirjoitus tapahtuu tägitaulun IO-osoitteiden kautta.

Käyttöönotto tapahtuu I-Deviceksi muutettavan logiikan asetuksissa aktivoimalla kohdassa ”PROFINET interface [x1] → Operating mode → ”IO Device”] (Kuva 6).

Kuva 6. I-Device aktivoiminen

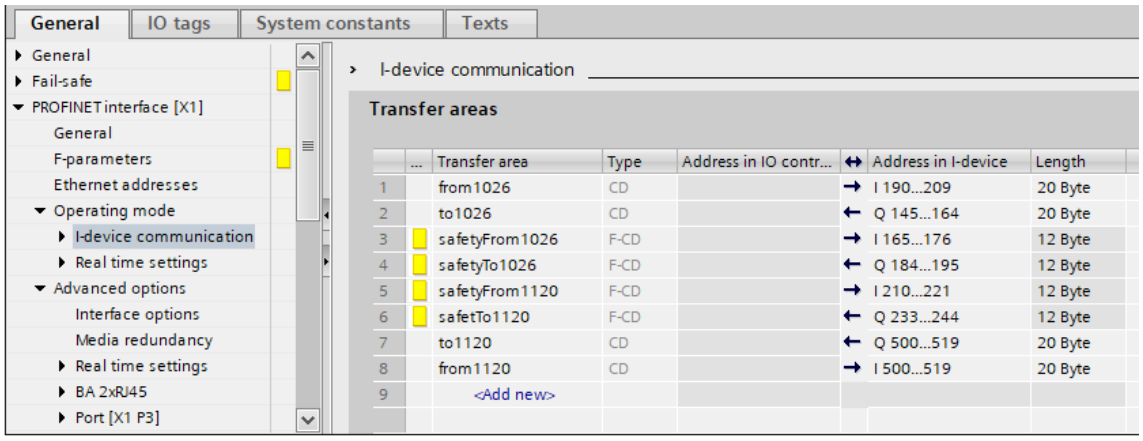


Tämän jälkeen alapuolelle aukeaa ”Transfer area” koskien haluttuja osoite-alueita, tässä kohtaa myös määritellään turvakommunikaation tarvitsemat/käyttämät turva-tulot ja -lähdöt. Kuvassa 7 I-Device on konfiguroitu kommunikoimaan kahden logiikan kanssa niin, että molemmille on luotu omat turva-IO:t sekä tavalliset IO:t.

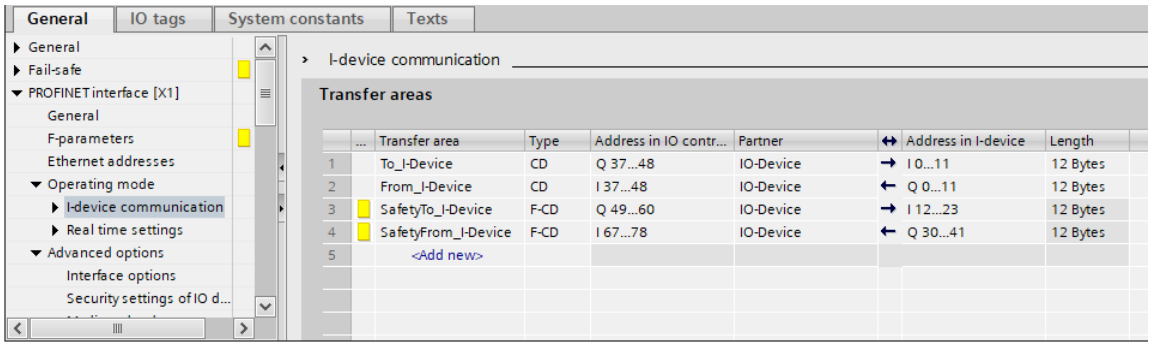
Kaikki kolme laitetta ovat omissa TIA-projekteissaan, jonka vuoksi ”Address in IO controller” on harmaana. Jos logiikat olisivat samassa projektissa, tässä ikkunassa näkyisi IO-logiikan tulot ja lähdöt, joita tulee käyttää (kuva 8).

Samassa määritellään myös käytettävän kaistan leveys. Kuvassa 7 kaistan leveydeksi on määritelty 20 tavua. Turvakommunikaatio käyttää aina 12 tavua ja sitä ei voi muuttaa.

Kuva 7. I-Devicen IO-alueet

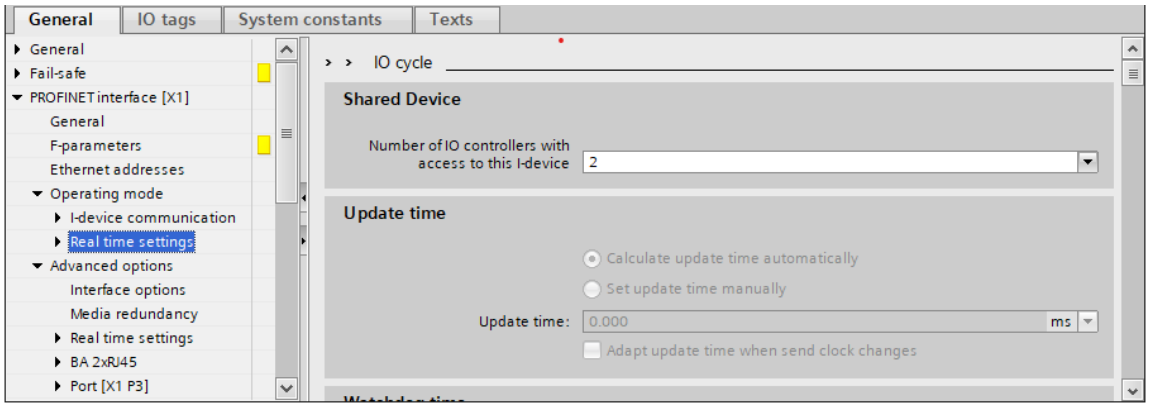


Kuva 8. I-Device ja IO Device samassa projektissa



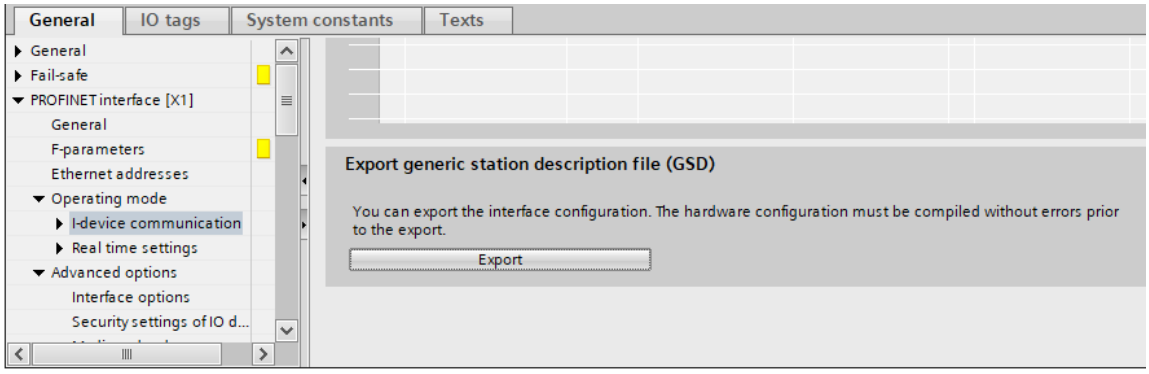
Real time settings-kohdassa (kuva 9) pitää vielä määritellä montako yhtäaikaista IO-laitetta I-Devicen voi olla saman aikaisesti yhteydessä. Tämä kannattaa pitää laitteistoa vastaavana määränä. Mikäli sallittujen laitteiden määrä on suurempi kuin käytettävä, voidaan I-Deviceen ottaa yhteys myös laitteistoon kuulumattomilla laitteilla, jolloin tietoturva heikkenee.

Kuva 9. Yhtäaikaisten IO laitteiden maksimimäärä I-Devicessä



Jos laitteet ovat eri projekteissa, edellä mainittujen määritysten jälkeen projekti pitää kääntää. Tämän jälkeen laitteesta luodaan Generic station description- eli GSD-tiedosto (kuva 10), tämän tiedoston voi tuoda muihin projekteihin ja I-Device näkyy tällöin projektissa tavallisena IO-laitteena.

Kuva 10. GSD-tiedoston luonti



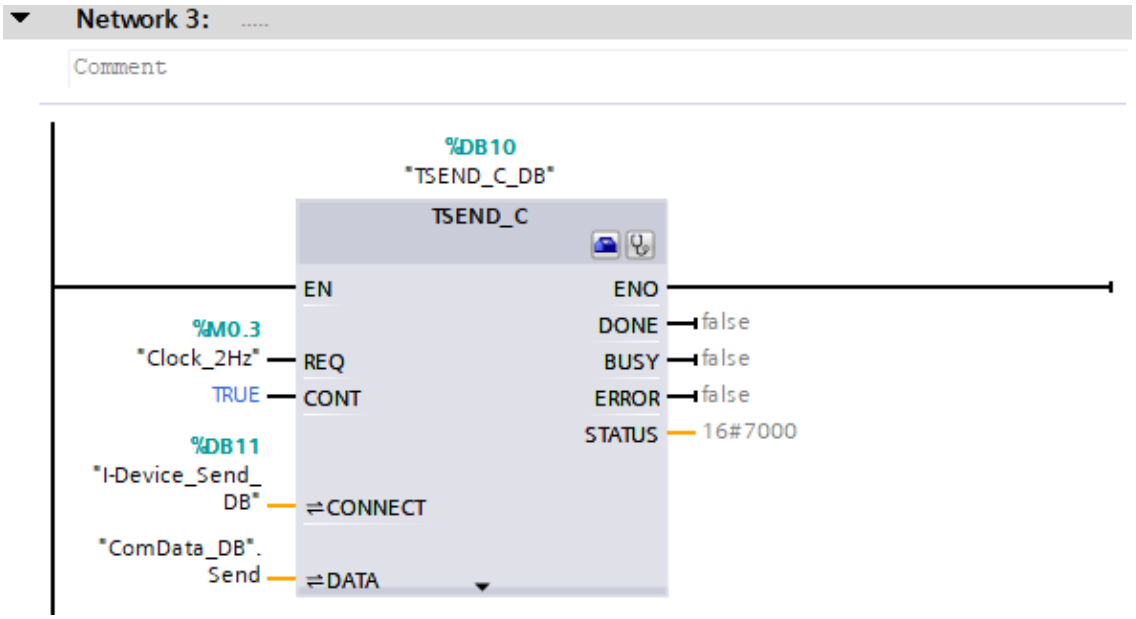
4.3 OUC

OUC eli Open User Communication menetelmässä yhteys määritetään parametroidulla tietorakenteella (esimerkiksi ISO-on-TCP), jonka jälkeen TSEND_C käynnistää lähetyksen ja TRCV_C vastaanoton.

4.3.1 TSEND_C-funktio

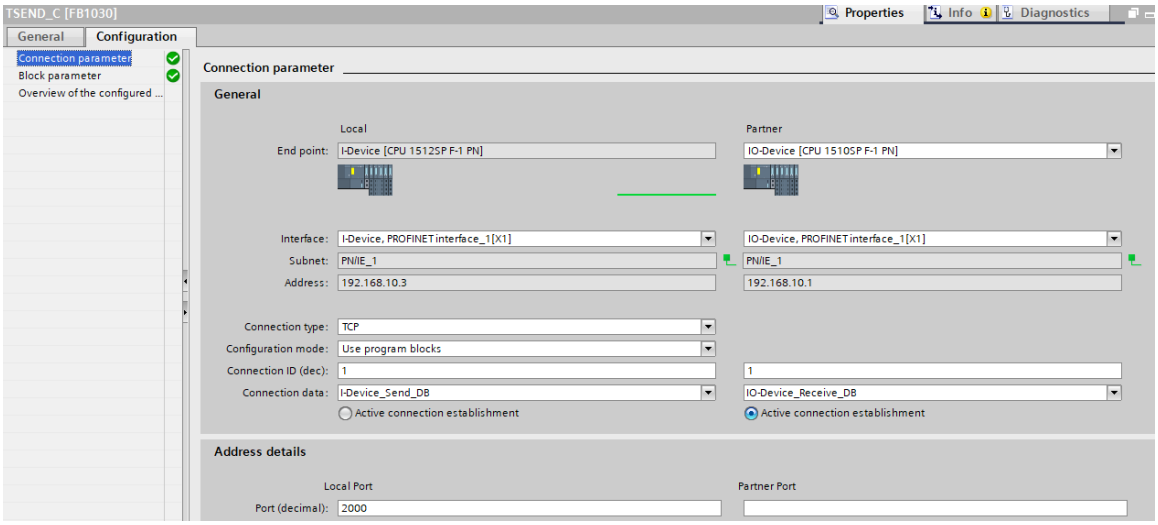
Yhteyden muodostamiseen tarvittavien parametrien syöttäminen tapahtuu avaamalla lohkon asetukset oikean ylälaidan sinisestä työkalulaatikon kuvasta (Kuva 11).

Kuva 11. TSEND_C funktio



Tämän jälkeen lohkon asetukset aukeavat, jossa varsinaisia yhteys-asetuksia voidaan muokata (Kuva 12).

Kuva 12. TSEND_C-funktion yhteys-asetukset



Samat asetukset määritellään myös vastaanottavan logiikan TRECV_C-funktioon. Huomioitavaa kuitenkin on se, että TCP/IP yhteydessä toinen osapuoli on aktiivinen ja toinen passiivinen. Passiivista yhteyttä pitävältä logiikalta pitää avata portti, jota pitkin aktiivinen puoli voi muodostaa yhteyden (Kuva 13).

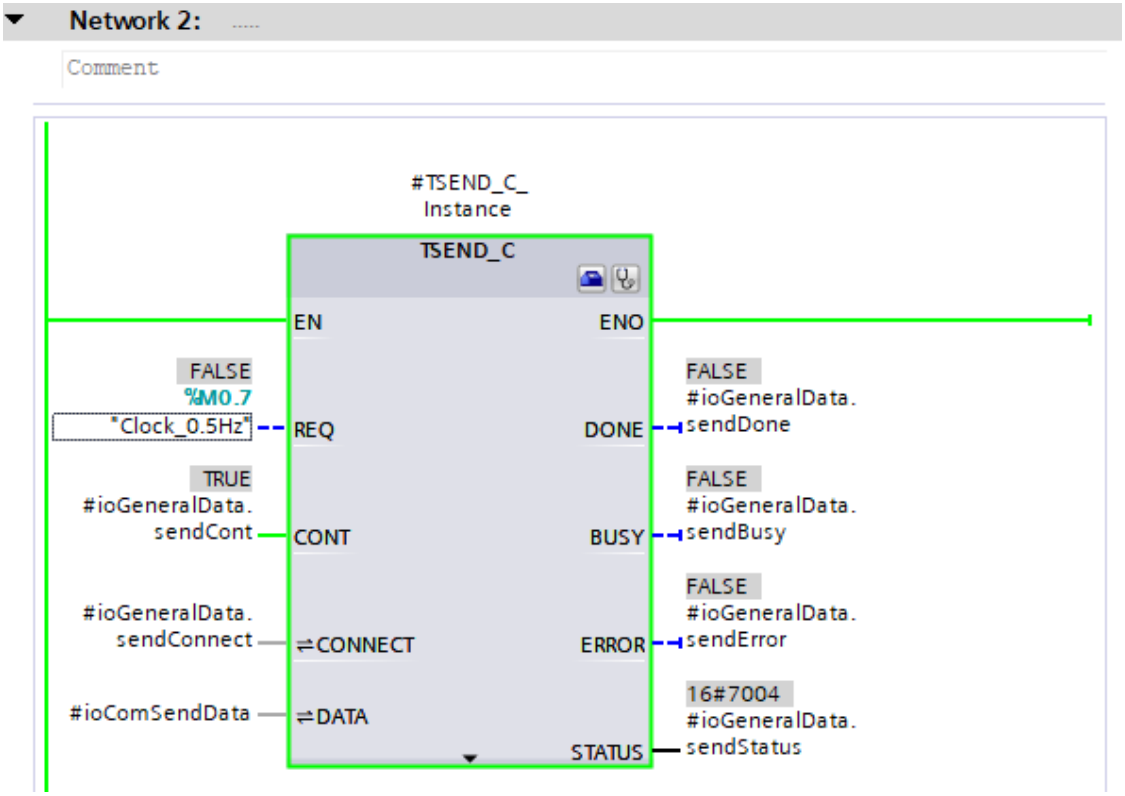
Kuva 13. Aktiivisena ja passiivisena toimivan logiikan valinta

Yhteyden tila määritellään CONT-tulon mukaan, 1 yhteyden automaattinen uudelleen luonti ja 0 yhteys ei käytössä. Varsinainen yhteyden muodostus tapahtuu TRCV_C-funktiossa.

Datan lähetys tapahtuu TSEND_C-funktion REQ-tuloon tuotavan bitin nousevalla reunalla. Tässäkin tavassa helpointa on käyttää logiikan kello-muistibitti, jolloin voidaan määritellä datan lähetysnopeus.

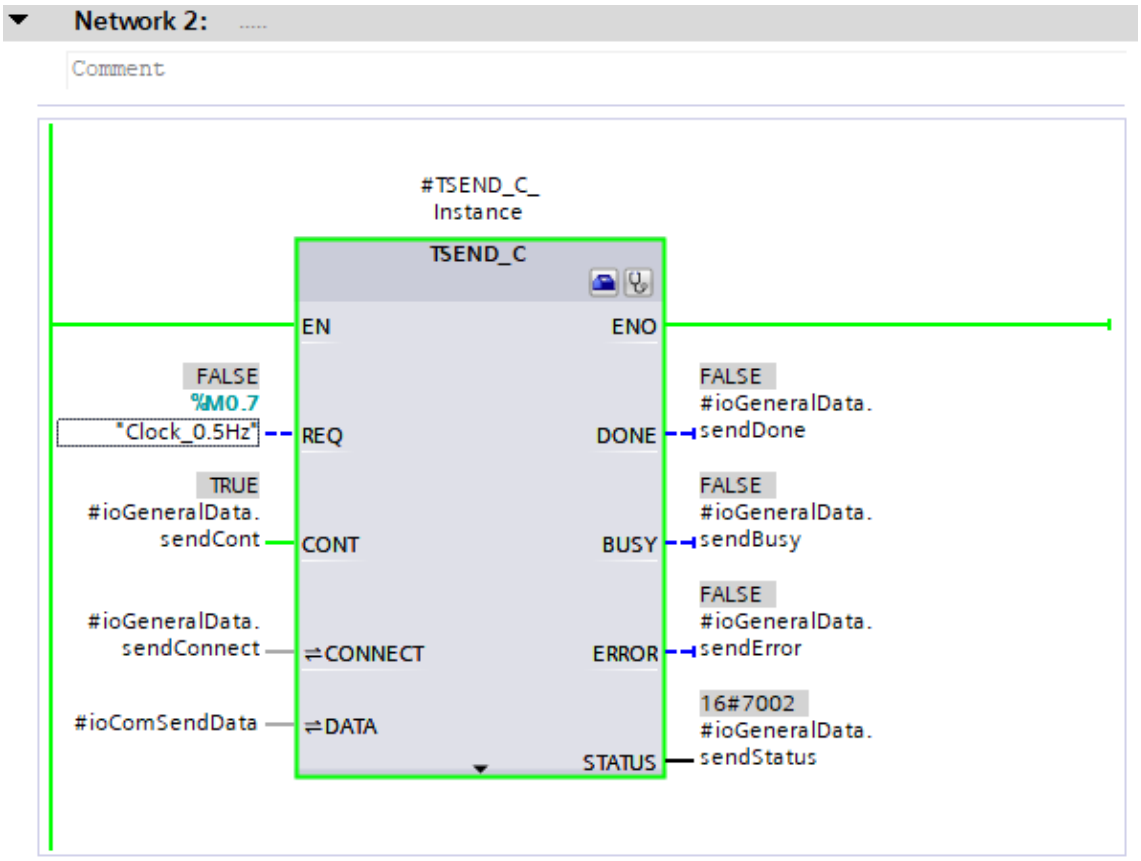
Yhteyden ollessa kunnossa lohkon STATUS-lähtö näyttää (16#7004) ja ERROR-lähtö on 0.

Kuva 14. TSEND_C yhteys kunnossa



Yhteyden katketessa STATUS-lähtö muuttuu (16#7002), tämä tarkoittaa, että CONT-tulon ollessa päällä funktio odottaa uutta yhteyttä vastapuolelta katkenneen yhteyden vuoksi (Kuva 15).

Kuva 15. TSEND_C yhteyden muodostamisen odotus

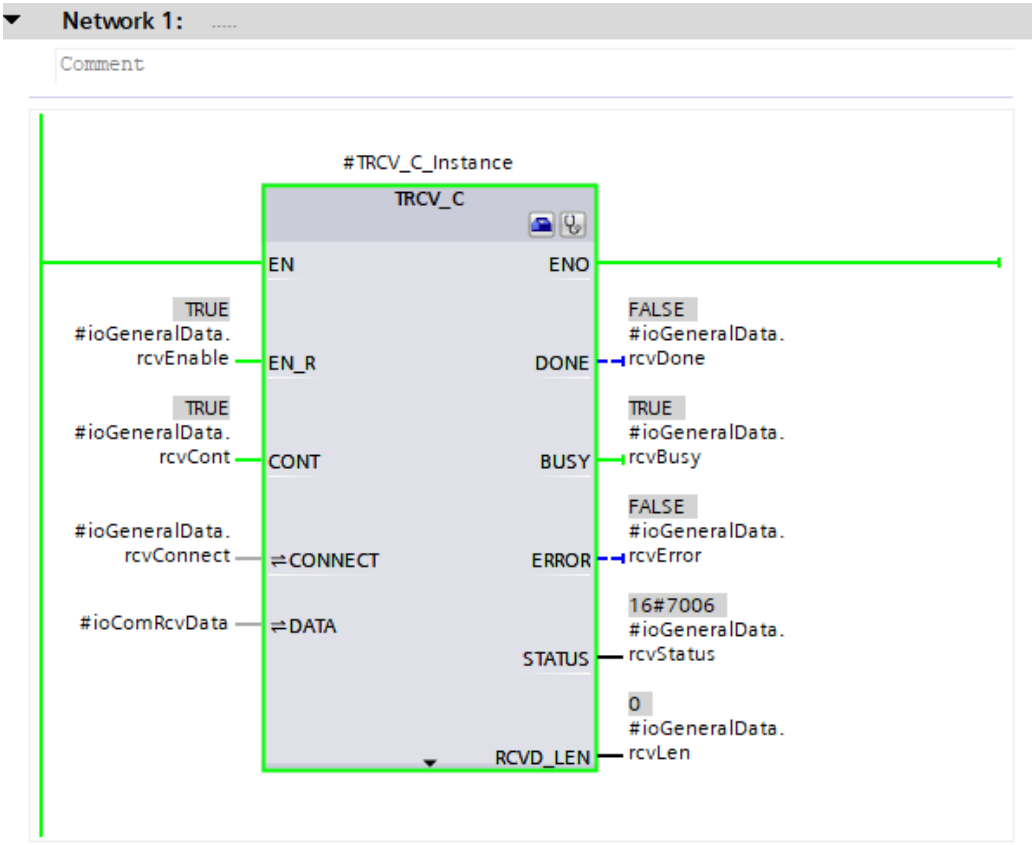


4.3.2 TRCV_C-funktio

TRCV_C funktion yhteys parametrisoidaan samoin kuin TSEND_C-fuktio, mutta funktioiden käytössä on kuitenkin eroa. TSEND_C funktiolla tuodaan vain lähetyspulssi ja kerrotaan, halutaanko yhteys muodostaa uudelleen katkeamisen jälkeen vai ei. TRCV_C-funktion CONT-tulo kertoo funktiolle, saako se muodostaa yhteyden vastapuoleen ja REQ-tulolla kerrotaan, saako lähetettävää dataa vastaanottaa.

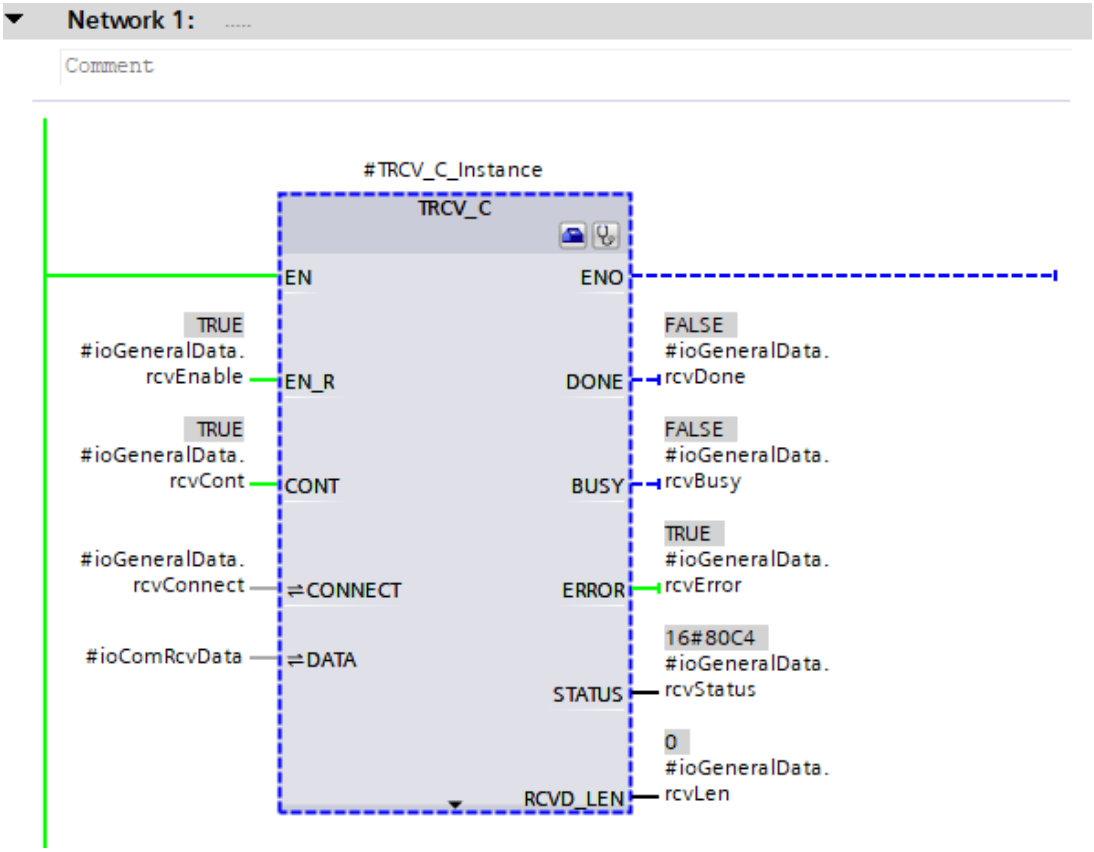
Yhteyden ollessa kunnossa TRCV_C-funktion STATUS-lähdössä on arvo (16#7006) ja BUSY-lähtö on päällä (Kuva 16).

Kuva 16. TRECV_C yhteys kunnossa



Yhteyden katketessa STATUS-lähtö muuttuu arvoon (16#80C4) ja ERROR-lähtö muuttuu arvoon 1 (Kuva 17). Kun CONT-tulo on edelleen päällä, funktio yrittää muodostaa yhteyden uudelleen automaattisesti.

Kuva 17. TRCV_C yhteys poikki



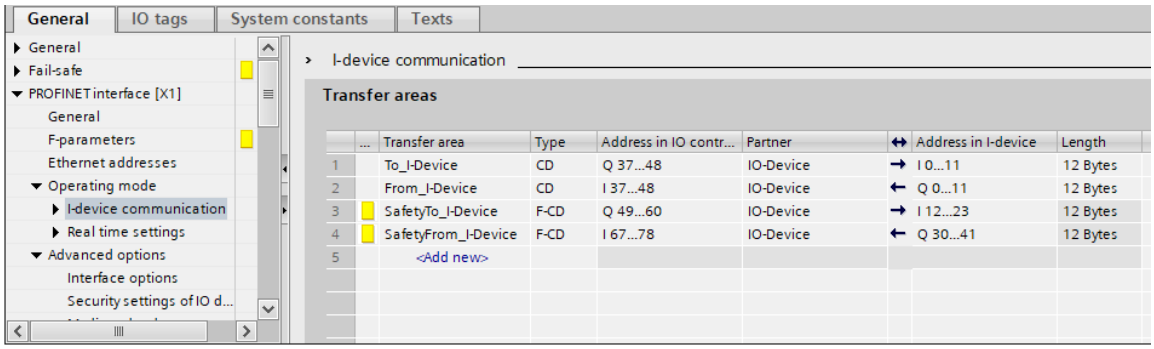
Yhteyden palatessa funktion arvot palautuvat automaattisesti takaisin samaan tilaan, kuin aiemmin mainitussa Kuvan 16 tekstissä on mainittu.

4.4 Turvakommunikaatio

Turvakommunikaatiossa nimensä mukaan kommunikoidaan logiikoiden välillä turvasignaaleilla, jolloin yhteyden katkeamista voidaan käyttää samoin kuin esimerkiksi hätä-seis painikkeen painamista. Yhteys pitää myös katkeamisen jälkeen kuitata samoin kuin muutkin turvaohjelmoinnin toiminnot.

Kommunikointi tapahtuu kahden funktion välityksellä SENDDP- ja RCVDP-funktiolla, näiden funktioiden käyttö varten pitää laitteet asettaa I-Device kommunikaatioon. Turvakommunikaatiota varten IO-listaan pitää lisätä erityiset SafetyIO:t. Tämä tapahtuu vaihtamalla Transfer areas-kohdan Type, muodosta CD, muotoon F-CD turvakommunikaatioon käytettävä datan pituus on aina kiinteä 12 tavua turvallisuuden varmistamiseksi (Kuva 18).

Kuva 18. Turvakommunikaation aktivointi



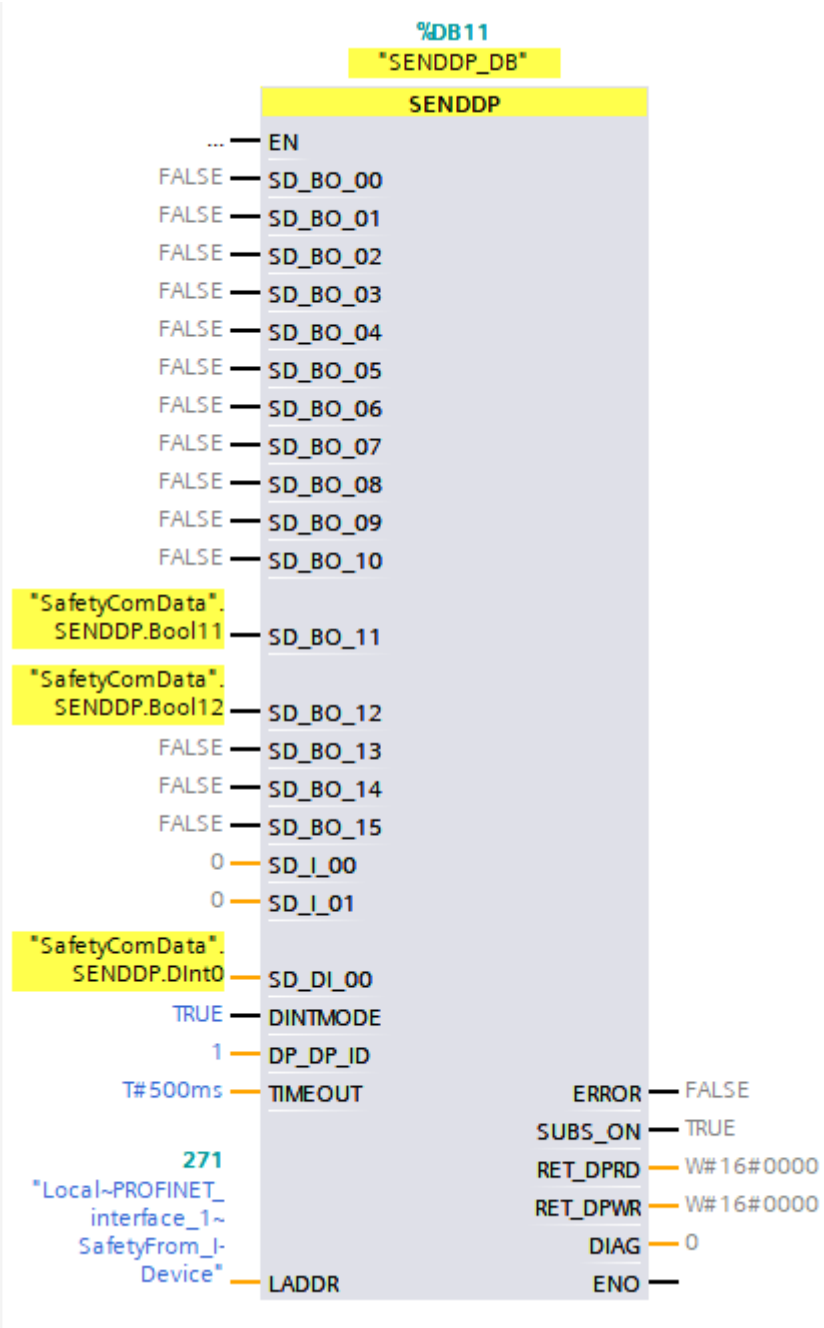
Tyypillisesti yksi viestipaketti koostuu 6 tavusta F-IO-dataa ja 6 tavusta F-parametrejä. Lähetettyyn dataan vastataan aina 6 tavun vahvistuksella. Funktiolla voidaan siis siirtää 16 yksittäistä bittiä (2 tavua) ja kaksi Int- tai Word-muuttujaa tietoa (2 tavua + 2 tavua), jos sovellutus vaatii siirrettäväksi näitä suurempaa arvoa, voidaan nämä kaksi Dataa yhdistää yhdeksi DInt-muuttujaksi (4 tavua). Tällöin SENDDP-funktiossa pitää valita DINTMODE-TRUE, nyt käytössä on SD_DI_00-tulo SD_I_00- ja SD_I_01-tulojen sijaan (Kuva 19).

Funktioiden DP_DP_ID-tulojen numeerinen arvo pitää olla sama, ajatellen kommunikaatiossa toimivia SENDDP-RCVDP-pareja. TIMEOUT-tulolla määritellään funktion yhteyden monitorointiaika, tämän ylittyessä yhteydessä tulkitaan olevan vikaa.

Funktioiden käytössä on myös huomioitavaa se, että muiden funktioiden käytössä ohjelmassa on suositeltavaa käyttää ohjelman alussa vastaanottavaa funktiota ja ohjelman lopussa lähettävää funktiota. Turvaohjelmoinnissa TIA-Portal ei anna lupaa käyttää SENDDP-, ja RCVDP-funktioita kuin turva-ohjelman alussa ja lopussa. RCVDP-funktion sisältävän Networkin pitää olla ohjelman ensimmäinen ja SENDDP-funktion sisältävä Network ohjelman viimeinen Network. Jos ohjelmassa tarvitaan useampi funktio, ne pitää sijoittaa omiin Networkeihinsa ohjelman alkuun ja loppuun.

4.4.1 SENDDP-funktio

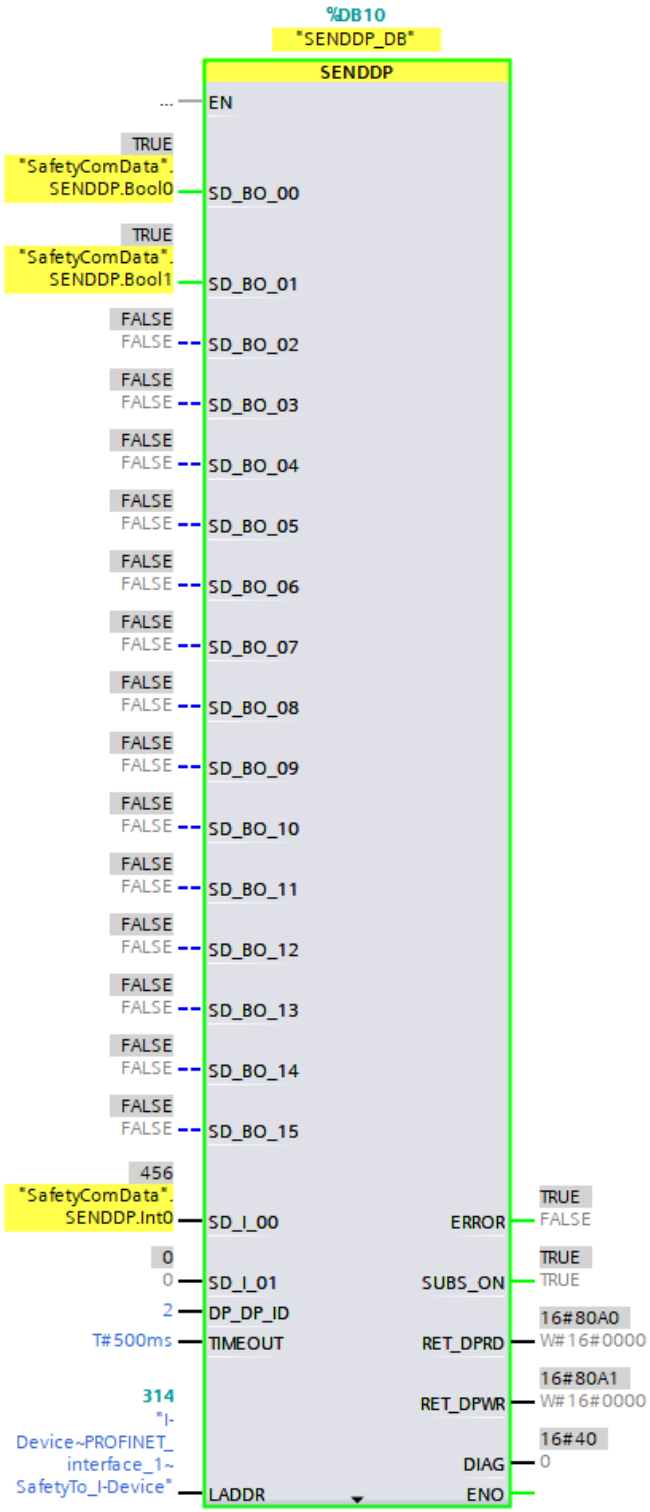
Kuva 19. SENDDP-funktio



Yhteyden ollessa kunnossa funktion ERROR- ja SUBS_ON- lähdöt ovat pois päältä ja RET_DRPD-, RET_DPWR- sekä DIAG-lähdöt ovat 0 (16#0000).

Yhteyden katketessa ERROR- ja SUBS_ON-lähdöt muuttuvat tilaan 1 ja RET_DRPD-, RET_DPWR- sekä DIAG-lähdöt antavat häiriökoodin (Kuva 20).

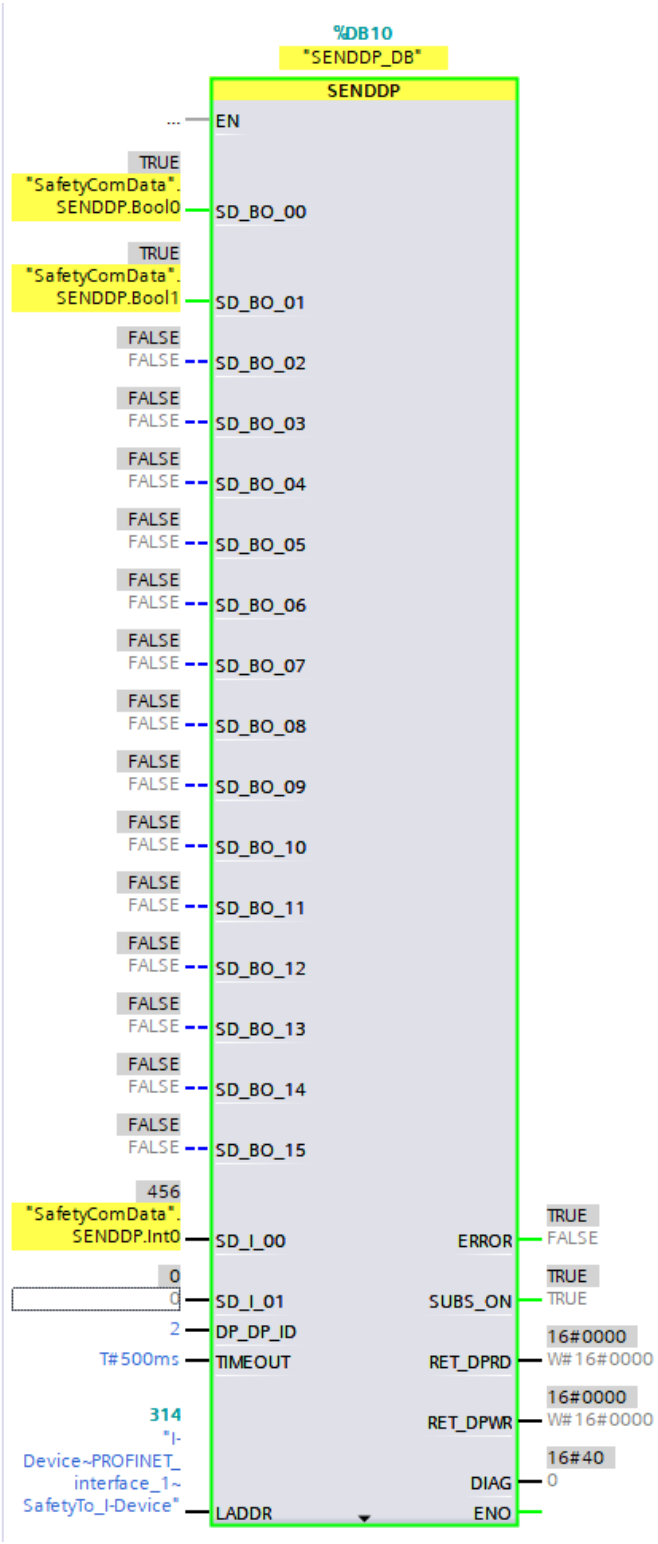
Kuva 20. SENDDP yhteys poikki



Yhteyden palatessa ja kommunikaation taas toimiessa funktion vikakoodit poistuvat ja asettuvat takaisin tilaan 0. ERROR-, ja SUBS_ON-lähdöt sen sijaan jäävät päälle merkiksi, että yhteys on taas kunnossa (Kuva 21).

Kun kyseessä on turvakommunikaatio, ohjelma ei lähde päälle enne kuin se on kuitattu samoin kuin muutkin turva-ohjelman toiminnot kuten hätä-seis-painike. Kommunikaation virheet kuitataan vastaanottavan logiikan RCVDP-funktiolla, ja SENDDP-funktio lähettävällä logiikalla odottaa.

Kuva 21. SENDDP yhteys palautunut, odottaa kuittausta

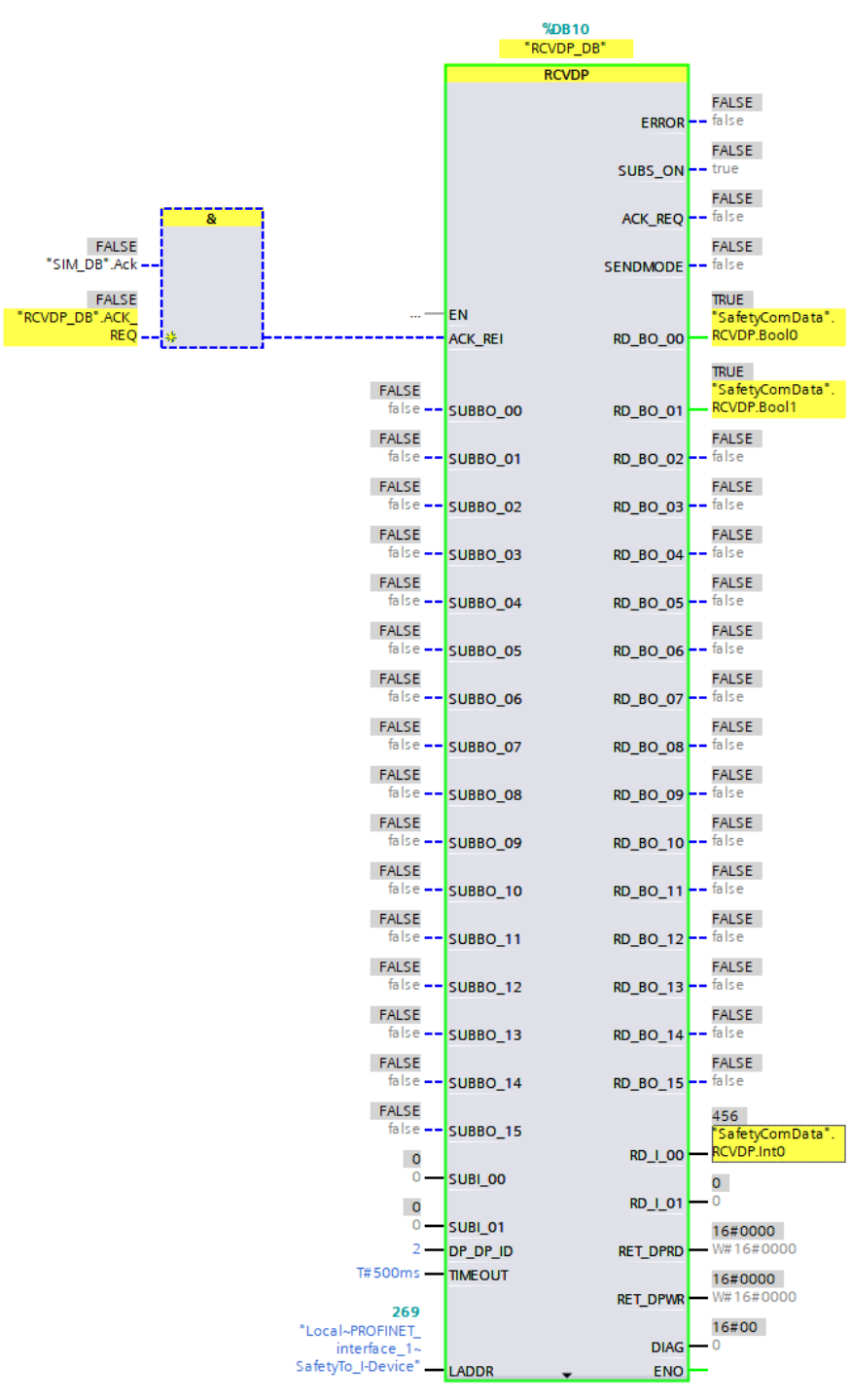


4.4.2 RCVDP-funktio

RCVDP-funktiossa on enemmän ominaisuuksia kuin SENDDP-funktiossa, joista suurimpana edellä mainittu yhteyden kuittaava ACK_REI-tulo (Kuva 22).

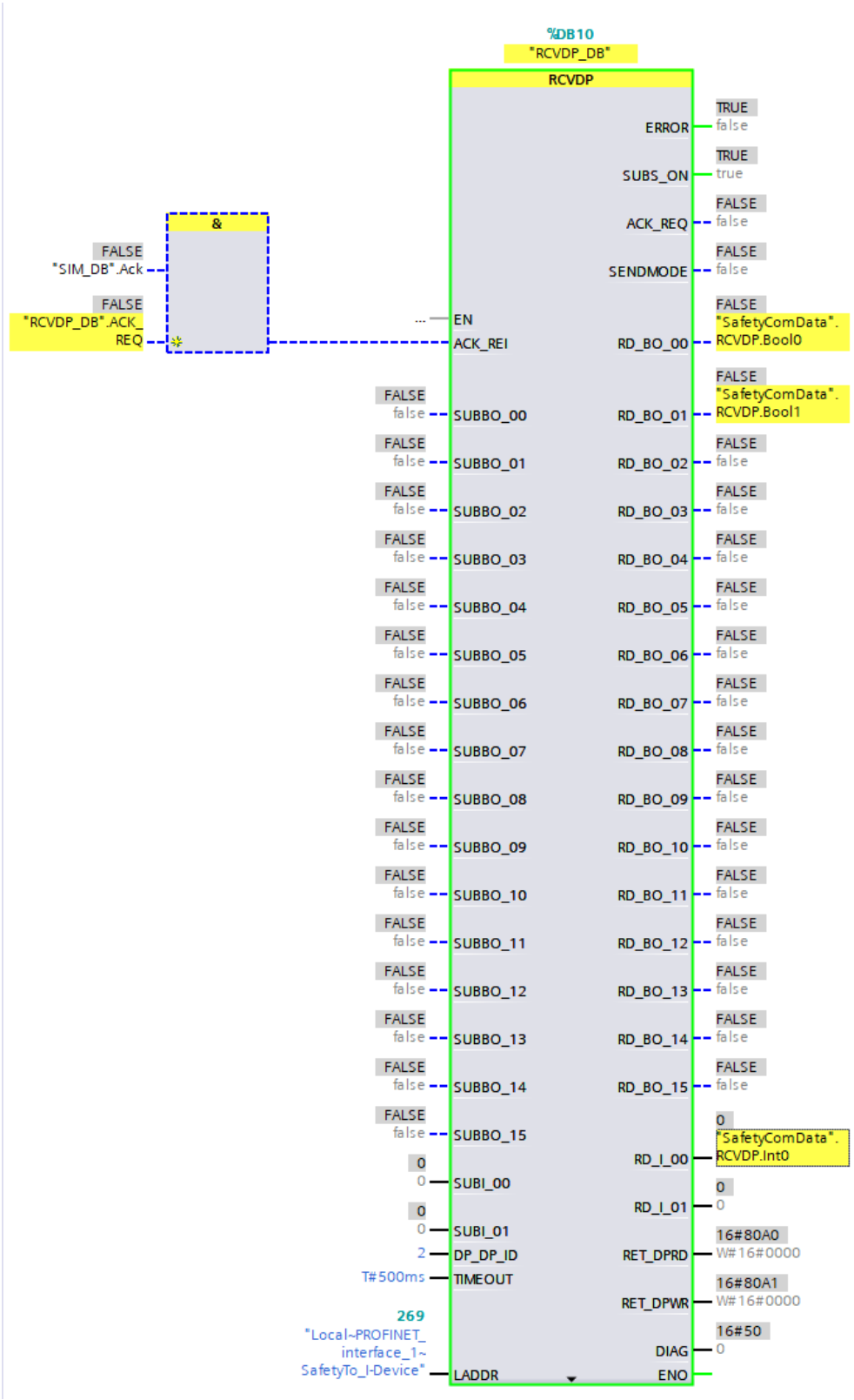
Samoin kuin SENDDP-funktiossa, kaiken ollessa kunnossa ERROR-, ja SUBS_ON-lähdöt ovat pois päältä myös virhekoodit ovat 0.

Kuva 22. RCVDP-funktio yhteys kunnossa



Yhteyden katketessa ERROR-, ja SUBS_ON-lähdöt aktivoituvat ja RET_DRPD-, RET_DPWR- sekä DIAG-lähdöt antavat virhekoodin (Kuva 23).

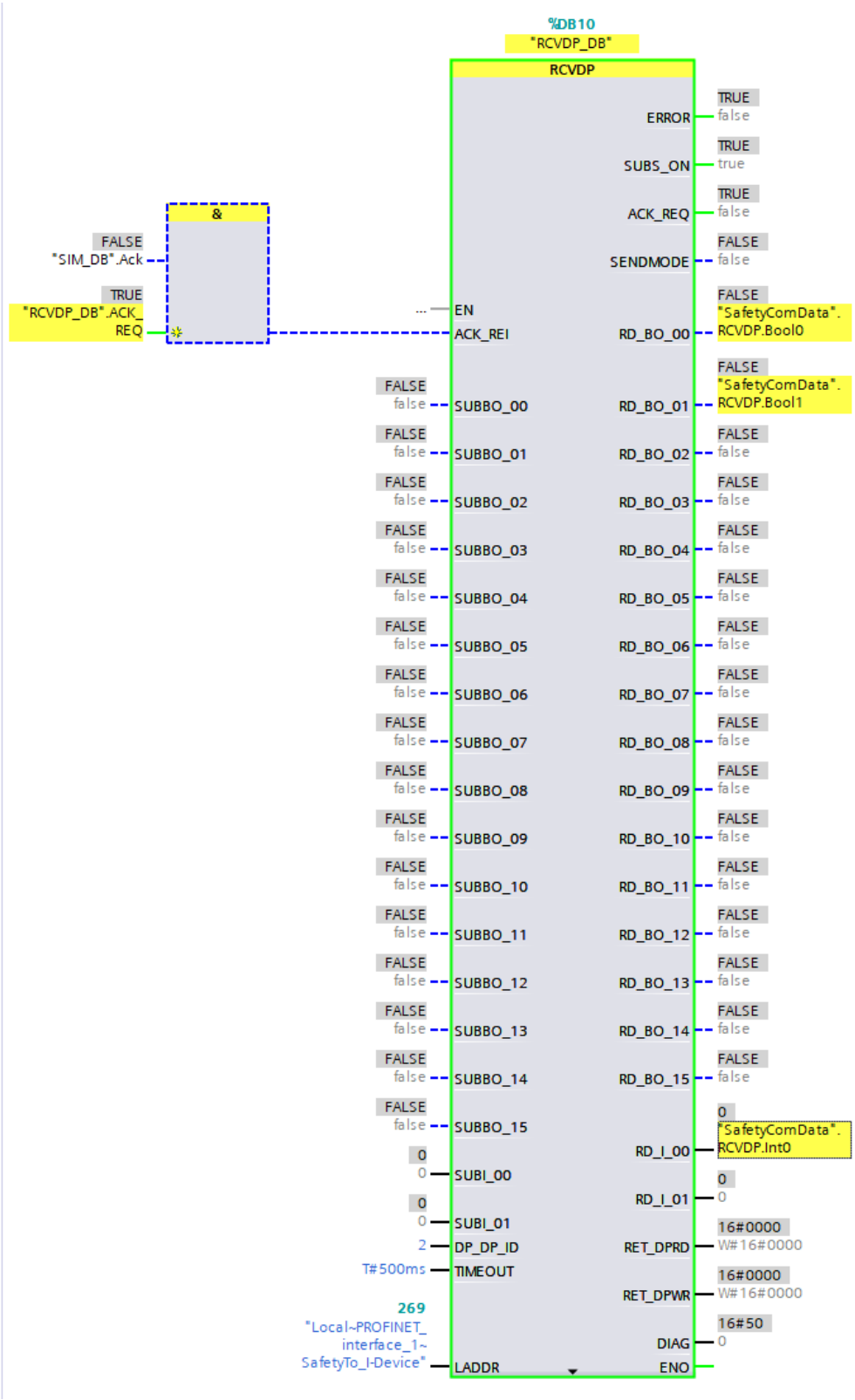
Kuva 23. RCVDP yhteys poikki



Kun yhteys on palautunut ja toiminnassa, RCVDP-funktio odottaa häiriön jälkeistä virheen kuittaamista (Kuva 24). Kuittaaminen tapahtuu ACK_REI-tuloon tuotavan bitin nousevalla

reunalla, tämän jälkeen ERROR-, ja SUBS_ON-lähdöt menevät pois päältä ja RET_DRPD-, RET_DPWR- sekä DIAG-lähdöt antavat koodin 0.

Kuva 24. RCVDP yhteys palautunut, odottaa kuittaamista



4.5 LCom -kirjasto

Tässä työssä esitellyistä kommunikaatiotavoista LCom-kirjasto on moderni, hyvin skaalautuva ja turvallinen datan siirtomenetelmä. Siinä kaikki toiminnot on yhdistetty yhteen ja samaan funktioon, jolloin käyttö on yksinkertaisempaa.

Kirjasto sisältää LCom_Communication -toimilohkon lisäksi LCom_typeConfig- ja LCom_typeDiagnostics- PLC datatyypit. Käyttöönotto tapahtuu luomalla datalohko, jossa on edellä mainituista datatyypeistä tehdyt taulukot (Kuva 25).

Kuva 25. LCom konfiguraatio- ja diagnostiikka-datalohko

6	configuration	"LCom_Config"	
7	connection	Struct	
8	interfaceID	HW_ANY	64
9	connectionID	CONN_ANY	16#0001
10	comService	USInt	1
11	isClient	Bool	FALSE
12	connectType	USInt	1
13	localPort	UInt	2000
14	partnerPort	UInt	2001
15	partnerIP	IP_V4	
16	ADDR	Array[1..4] of Byte	
17	ADDR[1]	Byte	16#C0
18	ADDR[2]	Byte	16#A8
19	ADDR[3]	Byte	16#0A
20	ADDR[4]	Byte	16#01
21	partnerQDN	String [254]	''
22	acceptUnknownPartner	Bool	FALSE
23	lifeSignCycleTime	Time	T#1S
24	sender	Struct	
25	timeSync	Struct	
26	diagnosticData	"LCom_Diagnostics"	
27	localConfig	Struct	
28	partnerConfig	Struct	
29	statistics	Struct	
30	bufferIndex	USInt	3
31	buffer	Array[0..63] of Struct	

Tässä työssä kommunikaatio tapahtuu TCP/IP-protokollan välityksellä, jolloin toinen logiikka toimii serverinä ja toinen clienttinä. Konfiguraatio- kohdassa määritellään kaikki yhteyden tarvitsemat asetukset. Samoin kuin OUC-kommunikaatiossa, joka perustuu myös TCP/IP protokollaan. Tällöin toinen logiikka asetetaan Clientiksi ja toinen serveriksi parametrillä ”isClient”. Kun logiikka valitaan Clientiksi, pitää myös ”partnerPort”-kohtaan laittaa serverin ”localPort”-kohtaan syötetty portin numero. Varmin tapa on laittaa molempiin kohtiin eri portin numerot, kuten kuvassa 25 on tehty.

DiagnosticDatan alta voidaan seurata diagnostiikkaa, esimerkiksi buffer-kohdassa näkyy aikaleima, virhekoodi sekä muuta tietoa (Kuva 26).

Kuva 26. Diagnostiikka buffer

26	diagnosticData	"LCom_Diagnostics"	
27	localConfig	Struct	
28	partnerConfig	Struct	
29	statistics	Struct	
30	bufferIndex	USInt	3
31	buffer	Array[0..63] of Struct	
32	buffer[0]	Struct	
33	status	Word	16#8604
34	timestamp	DTL	DTL#2025-0...
35	YEAR	UInt	2025
36	MONTH	USInt	4
37	DAY	USInt	24
38	WEEKDAY	USInt	5
39	HOUR	USInt	15
40	MINUTE	USInt	50
41	SECOND	USInt	53
42	NANOSECOND	UDInt	922444272
43	isActive	Bool	FALSE
44	subFunctionErrorID	Word	16#8086
45	additionalValue1	Real	0.0
46	additionalValue2	Real	0.0
47	additionalValue3	Real	0.0
48	additionalValue4	Time	T#0MS

LCom-kommunikaation siirtämä data poikkeaa myös muiden sovellutusten tavasta, siinä missä muissa kommunikaatioissa siirretään datalohkon sisältö tietyn määrän verran ja tietotyypillä ei ole merkitystä, LCom-funktiolla siirretään aina taulukko tavuja (Array [0..15] of tavua) (Kuva 27). Taulukon pituus voidaan määritellä halutun mittaiseksi (maksimi 4 294 967 295 tavua) eli funktiolla voidaan siirtää hyvin suuria määriä dataa.

Kuva 27. LCom lähetys bufferi

5			sendBuffer	Array[0..15] of Byte	
6			sendBuffer[0]	Byte	16#0
7			sendBuffer[1]	Byte	16#0
8			sendBuffer[2]	Byte	16#0
9			sendBuffer[3]	Byte	16#0
10			sendBuffer[4]	Byte	16#0
11			sendBuffer[5]	Byte	16#0
12			sendBuffer[6]	Byte	16#0
13			sendBuffer[7]	Byte	16#0
14			sendBuffer[8]	Byte	16#0
15			sendBuffer[9]	Byte	16#0
16			sendBuffer[10]	Byte	16#0
17			sendBuffer[11]	Byte	16#0
18			sendBuffer[12]	Byte	16#0
19			sendBuffer[13]	Byte	16#0
20			sendBuffer[14]	Byte	16#0
21			sendBuffer[15]	Byte	16#0

Kun asetukset, diagnostiikka ja taulukko lähetettävälle-, sekä vastaanotettavalle datalle on luotu, ne voidaan yhdistää varsinaiseen LCom_Communication-toimilohkoon (Kuva 28).

Käyttö perustuu samaan kuin muilla funktioilla, enable-tulolla määritetään, halutaanko kommunikaatiota käyttää ja send-tulon nousevalla reunalla lähetetään sendBuffer-tuloon tuotu lähetettävän datan taulukko.

Muista tavoista poiketen LCom-funktiolle kerrotaan, kuinka pitkä pätkä taulukosta halutaan lähettää, sendDataLength-tuloon annetaan arvo, kuinka monta tavua taulukon alusta lähetetään. Jättämällä arvon tyhjäksi, tulee oletusarvo 16#FFFF_FFFF (4 294 967 295 tavua) silloin funktio lähettää koko luodun taulukon vastaanottajalle.

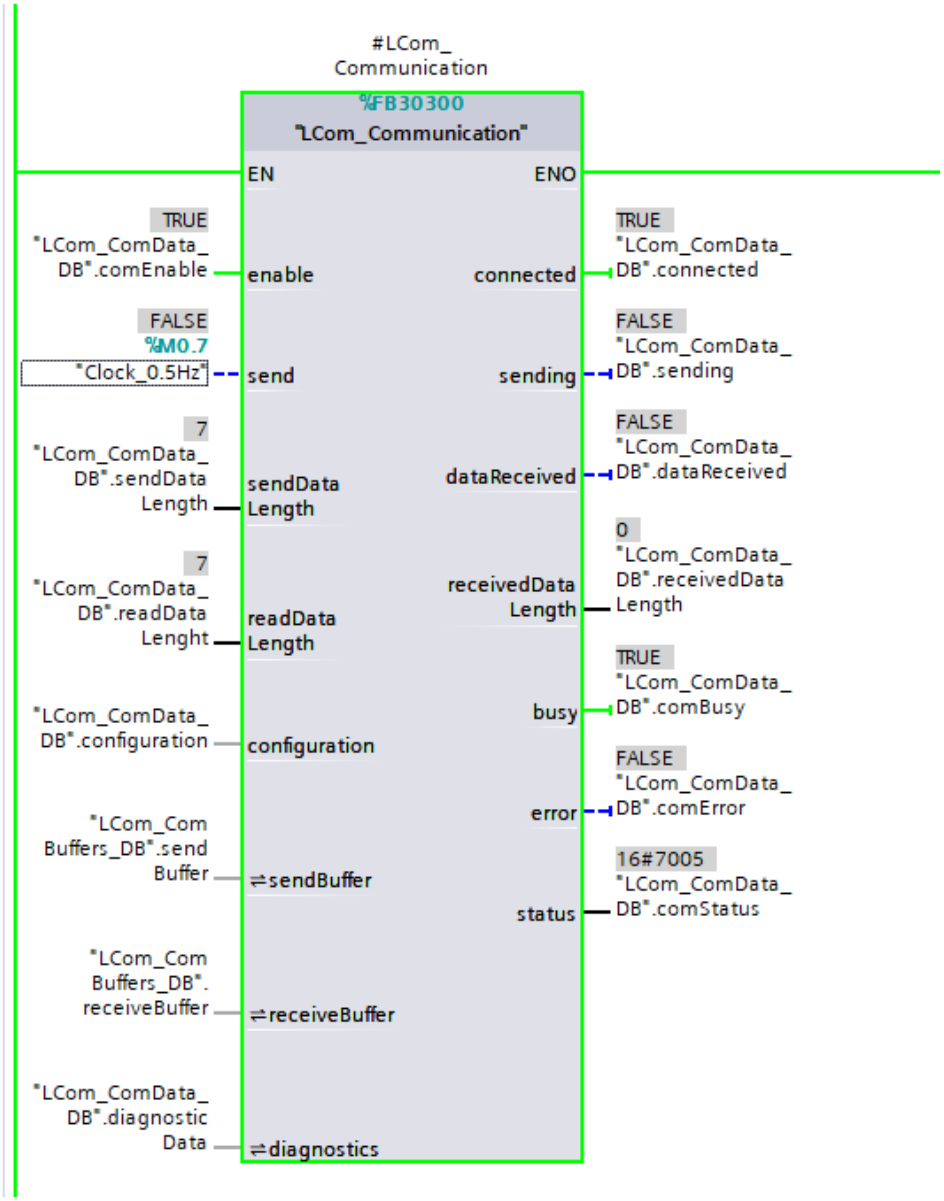
Kun halutaan määritellä saapuvan datan pituus, kirjataan sen pituus kohtaan readDataLength. Jos tuloon ei määritellä pituutta funktio odottaa aina vastaanottavansa maksimipituisen datan eli 4 294 967 295 tavua. Tällöin lähetetty data vastaanotetaan, mutta ei voida kertoa dataReceived-lähdöllä onko kaikki data vastaanotettu. receivedDataLength-tulo näyttää viimeisimpänä saapuneen datan pituuden.

Jotta dataReceived-tieto saadaan, pitää saapuneen datan pituus määritellä oikean pituiseksi readDataLength-tuloon. Tällä määritellään myös vastaanotettavan datan

maksimipituus. Eli jos lähetetty data on pidempi, siitä ei vastaanoteta kuin alun määritelty pituus.

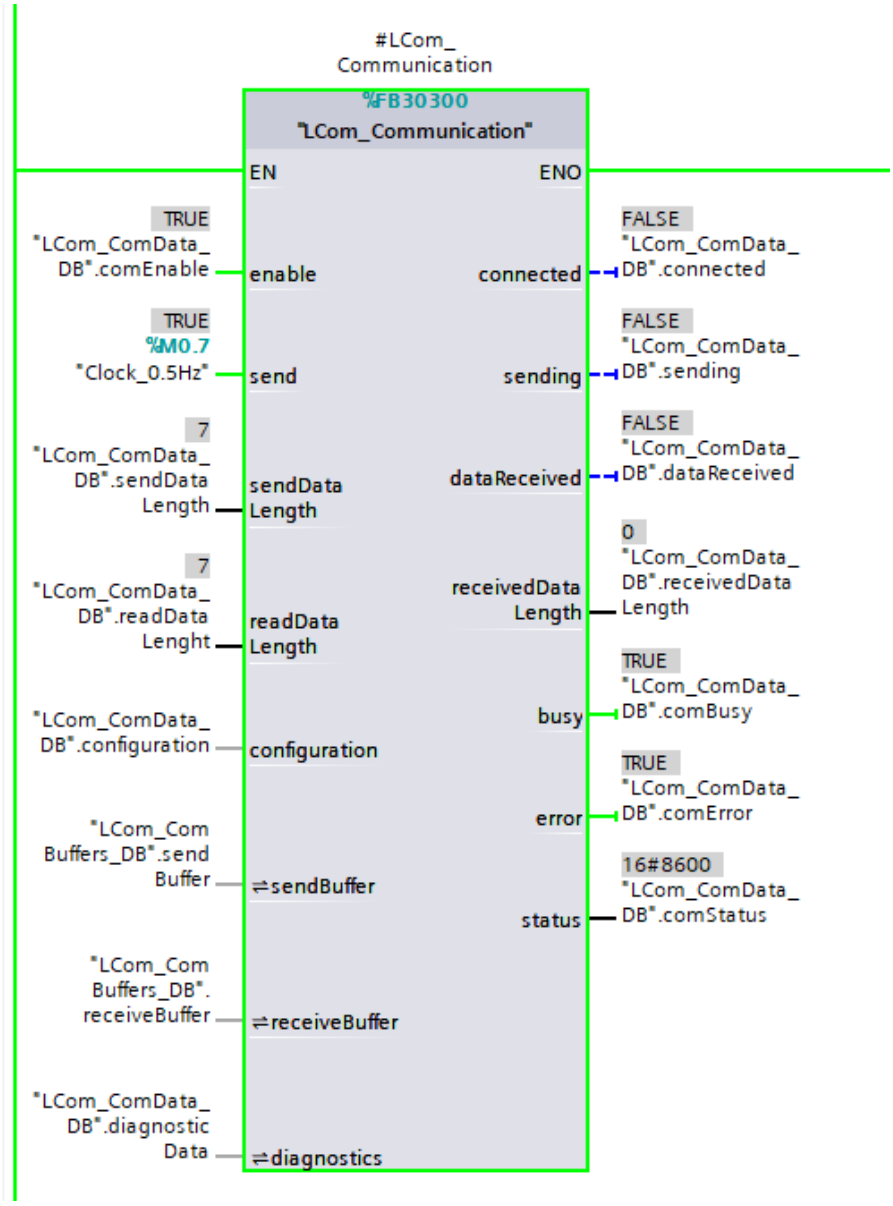
Kun enable-tulon on 1 ja yhteys on kunnossa connected-, sekä busy-lähdöt ovat päällä ja status on 16#7005 (Kuva 28).

Kuva 28. LCom-funktio, yhteys kunnossa



Yhteyden katketessa connected-lähtö saa arvon 0, error arvon 1 ja status arvo muuttuu tässä tapauksessa arvoon 16#8600 (Kuva 29).

Kuva 29. LCom-funktio yhteys poikki



5 Tulokset

Työn aikana toteutettiin viisi erilaista tiedonsiirtomenetelmää kahden Siemensin S7-1500-sarjan logiikan välille PROFINET-verkon kautta. Jokainen menetelmä rakennettiin samaan laitteistoympäristöön, mikä mahdollisti reilun ja vertailukelpoisen arvioinnin. Menetelmät erosivat toisistaan merkittävästi niin käyttöönoton, toimintavarmuuden kuin ohjelmointityön määrän suhteen.

PUT/GET osoittautui suoraviivaiseksi ja nopeasti käyttöönotettavaksi ratkaisuksi. Yhteys saatiin muodostettua muutamalla asetuksella ja saatiin käyntiin ilman ohjelmointia toisessa logiikassa. Käytön helppous oli vahvuus, mutta samalla huomattiin menetelmän heikkoudet: se ei tarjoa kunnollista virheentunnistusta eikä turvallisuusmekanismeja, mikä rajoittaa sen käyttöä erityisesti kriittisissä sovelluksissa.

I-Device-menetelmä erottui edukseen siinä, miten yksinkertaisesti ja selkeästi logiikat voitiin liittää toisiinsa IO-laitteina. Tämä mahdollisti deterministisen ja reaaliaikaisen tiedonsiirron ilman erillisiä ohjelmalohkoja. I-Device on erityisen hyödyllinen tilanteissa, joissa useat logiikat kommunikoivat keskenään vakioidulla rakenteella.

Open User Communication, eli OUC, tarjosi käyttäjälle paljon vapauksia: yhteyksiä voitiin rakentaa käsin ja tiedonsiirto määritellä tarkasti halutulla tavalla. Tämä mahdollisti monimutkaisempien tiedonsiirtoskenaarioiden toteuttamisen, mutta toisaalta lisäsi myös työn määrää ja virheiden mahdollisuutta, mikä näkyi varsinkin yhteydenhallinnan ja virheenkäsittelyn toteutuksessa.

Turvakommunikaatio toimi odotetusti. Yhteys vaatii F-tyypin logiikat, mutta kun nämä ovat kunnossa pystyttiin toteuttamaan turvallinen, standardien mukainen viestintä ilman ylimääräisiä kaapeleita tai laitteita. Viestintä toimi vakaasti ja virheenkäsittely oli automatisoitu, mikä paransi järjestelmän kokonaisturvallisuutta.

LCom-tiedonsiirto perustui valmiiseen funktiolohkoon, jolla muodostettiin TCP-yhteys logiikoiden välille. Käyttöönotto oli helppoa: clientille ja serverille määriteltiin toistensa IP-osoitteet ja clientille vielä portin numero, tämän jälkeen serveri hyväksyi yhteyden automaattisesti. Yhteydenhallinta, virheiden käsittely ja uudelleenyhdistäminen tapahtuivat taustalla ilman, että käyttäjän tarvitsi puuttua asiaan. Lisäksi lohko tarjosi selkeän diagnostiikan, josta näki suoraan yhteyden tilan. LCom suoriutui erinomaisesti myös yhteyskatkosten simuloinneissa.

6 Johtopäätökset

Kaikki toteutetut viestintämenetelmät toimivat teknisesti hyvin, mutta niillä oli selkeitä eroja sen suhteen, missä tilanteissa niitä kannattaa käyttää. Yhtä parasta menetelmää ei ole, vaan oikea valinta riippuu käyttökohteesta ja sen vaatimuksista.

Jos tavoitellaan mahdollisimman nopeaa ja kevyttä käyttöönottoa, ilman suuria turvallisuusvaatimuksia, PUT/GET on hyvä vaihtoehto. Se toimii erinomaisesti yksinkertaisissa järjestelmissä, joissa nopeus ja helppous ovat tärkeämpiä kuin virheentunnistus.

I-Device on erinomainen ratkaisu moniohjainjärjestelmiin, joissa logiikoiden välistä tiedonsiirtoa halutaan toteuttaa ilman ylimääräistä ohjelmointia. Se on myös luotettava ja selkeä toteuttaa. Jos laitteistossa käytetään myös turvakommunikaatiota, I-Device on hyvä valinta sillä kommunikaatiotavat käyttävät samaa laitteiston määrittelyä.

OUC tarjoaa joustavuutta ja ohjelmoitavuutta niille, jotka tarvitsevat tarkempaa kontrollia yhteyksien hallintaan tai haluavat rakentaa omaan sovellukseen sopivan protokollan. Se vaatii enemmän suunnittelua ja kokemusta, mutta toimii hyvin silloin kun yksinkertaiset ratkaisut eivät riitä.

Turvakommunikaatio puolestaan on lähes itsestään selvä valinta turvallisuuskriittisiin kohteisiin. Se tuo järjestelmään standardien mukaisen luotettavuuden ja helpottaa konfigurointia, kun käytössä ovat sopivat F-tyypin CPU:t.

Lopuksi LCom nousi esiin erityisesti käytettävyyden ja toimintavarmuuden osalta. Se tarjosi hyvin toimivan TCP-yhteyden logiikoiden välille. Lohkon automaattinen yhteydenhallinta ja hyvä diagnostiikka vähensivät virheiden mahdollisuutta ja helpottivat ylläpitoa. LCom on hyvä valinta silloin, kun tarvitaan yksinkertainen mutta toimintavarma tapa siirtää tietoa kahden logiikan välillä TCP:n avulla.

Yhteenvetona voidaan todeta, että viestintämenetelmää valittaessa kannattaa aina miettiä järjestelmän vaatimukset, oma osaaminen sekä ylläpidettävyyden ja tietoturvan tarpeet. Tämä työ tarjoaa käytännönläheisen katsauksen siihen, mitä eri vaihtoehtoja on ja mihin ne parhaiten soveltuvat.

Lähteet

Siemens AG. (11/2022). PROFINET with STEP 7, Function Manual.

Siemens AG. SIMATIC (2022a). I-Device Configuration and Use, Function Manual.

Siemens AG. SIMATIC (06/2024). LCom Libraries Communication Controller.

Siemens AG. SIMATIC (2022b). Open User Communication (OUC), Function Manual.

Siemens AG. SIMATIC (2022c). PUT/GET Communication via PROFINET, Function Manual.

Siemens AG. SIMATIC (2022d). Safety Communication, Function Manual