Tampereen ammattikorkeakoulu

Sähkösuunnittelun laite-määrittelyn tehostaminen

Lasse Nykänen

OPINNÄYTETYÖ
Huhtikuu 2020

Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikan tutkinto-ohjelma
Automaatiotekniikka

NYKÄNEN, LASSE:
Sähkösuunnittelun laitemäärittelyn tehostaminen

Opinnäytetyö 33 sivua, joista liitteitä 2 sivua
Huhtikuu 2020

Opinnäytetyönä tehtiin Microsoft Office Excel -taulukko-ohjelmassa ohjelmoitu työkalu, jonka avulla voidaan luoda kytkentälistoja ohjelmoitaville logiikoille automaatioympäristössä. Työkalu ohjelmoitiin Microsoft Officen kehittämällä Visual Basic for Applications -ohjelmointikielellä, jonka avulla on mahdollista tehdä erilaisia toimintasarjoja suorittavia ohjelmia.

Opinnäyteyön teettäjänä toimi Tampereella sijaitseva Insta Automation Oy, joka toimii pääasiassa teollisuusautomaation parissa. Työn tarkoitus on helpottaa, nopeuttaa ja tehostaa automaatiosuunnittelijan työtä kytkentälistojen tekemisessä sekä yhdenmukaistaa kytkentälistoja mahdollisimman paljon. Tavoitteena on myös pyrkiä vähentämään inhimillisten virheiden määrää automatisoimalla kytkentälistojen luomisessa tapahtuvat useasti toistuvat toimenpiteet sekä muistuttaa suunnittelijaa tarvittavista kytkennöistä.

Opinnäytetyönä tehty työkalu koostuu eri valmistajien logiikkalaitteita ja logiikkakortteja sisältävästä kirjastosta. Työkalun avulla suunnittelijan valitsemien logiikkalaitteiden ja -korttien kytkentäpisteet luodaan Excel-taulukkoon automaattisesti, eikä suunnittelijan tarvitse tehdä työtä manuaalisesti itse. Työkaluun ohjelmoitiin myös kattava kirjasto teollisuusautomaatiossa yleisimmin käytetyistä kenttälaitteista.

Työn aikana yrityksen kokeneemmilta suunnittelijoilta kysyttiin laajalti mielipiteitä valmiin työkalun toimintoihin liittyen. Näin ollen lopullista työkalua käyttävät suunnittelijat saivat mahdollisuuden vaikuttaa työn lopputulokseen, joka puolestaan vaikuttaa positiivisesti valmiin työkalun käyttöasteeseen. Jatkossa työkalua on mahdollista kehittää ohjelmoimalla siihen lisää eri valmistajien logiikkalaitteita sekä lisäämällä valmiiksi ohjelmoituja kenttälaitteita tarpeiden mukaan.

Opinnäytetyössä tehtyä ohjelmakoodia ei tässä raportissa esitetä kokonaisuudessaan, koska se on yrityksen pyynnöstä julistettu salaiseksi.

Asiasanat: sähkösuunnittelu, Excel, VBA-ohjelmointi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Electrical and Automation Engineering
Automation Engineering

NYKÄNEN, LASSE:
Improving the Efficiency of Device Specification in Electrical Engineering

Bachelor's thesis 33 pages, appendices 2 pages
April 2020

The purpose of this thesis was to design a useful tool to create connection lists for programmable logics in the field of automation. The tool was programmed in Microsoft Office's Visual Basic for Applications programming language. By using VBA language it is possible to create programs which run different kinds of actions in Excel.

The thesis was commissioned by Insta Automation Inc, which is mainly focused in industrial automation. The main objective of this thesis was to make it possible for automation engineers to make connection lists in an easier, faster and more efficient way. The goal was also to reduce the number of human errors by automating repeated operations and reminding the designer of the necessary connections.

The tool contains a library of logic devices and logic cards from different manufacturers. The tool automatically creates connection points for logic devices and cards selected by the designer, removing the need to do the work manually. A comprehensive library of commonly used functionalities in industrial automation was also included in the tool.

Experienced designers in the company were asked for opinions about the functionality of the finished tool during this study. In this way, designers who will use the final version of the tool were given the opportunity to influence the outcome of the work, which affects positively the utilization rate of the finished tool. In the future, it will be possible to develop the tool by including more logic devices from different manufacturers and adding functionalities as needed.

The program code made in this thesis is not fully presented in this report because it has been declared confidential at the request of the company.

Key words: electrical engineering, Excel, VBA programming

SISÄLLYS

LYHENTEET JA TERMIT

AI	Analog Input. Analoginen tulotieto
AO	Analog Output. Analoginen lähtötieto
CPU	Central Prosessing Unit. Logiikkalaite, joka suorittaa siihen ohjelmoituja käskyjä ja hallitsee siihen liitettyjä kenttälaitteita
DI	Digital Input. Binäärinen tulotieto
DO	Digital Output. Binäärinen lähtötieto
HW	Hardware. Tarkoittaa käsiteltävän kohteen fyysistä laitteistoa. HW-suunnitteluun sisältyy kaikki laitteiston toimimiseen tarvittavien komponenttien määritys.
I/O	Input/Output. Nimitys automaatiossa käytetyille tulo- ja lähtötietopisteille
Makro	Halutun toimintasarjan suorittava ohjelma
SW	Software. Tarkoittaa ohjelmistoa. SW-suunnittelulla tarkoitetaan ohjelmistojen suunnittelua ja koodaamista.
VBA	Visual Basic for Applications. Ohjelmointikieli, jota käytetään muun muassa Microsoft Excel -ympäristössä

1 JOHDANTO

Opinnäytetyön tarkoitus on kehittää Microsoft Office Excel -ympäristössä työkalu, joka helpottaa ja nopeuttaa automaatiosuunnittelijan työtä ja vähentää inhimillisten virheiden määrää. Automaatioprojektien yksi oleellisimpia vaiheita on logiikkajärjestelmien tulo- ja lähtötietopisteiden suunnittelu sekä kytkentäpisteiden määrittely. Tulo- ja lähtötietopisteistä ja niihin kytkettävistä kenttälaitteista laadittava I/O-lista tehdään jo varhaisessa vaiheessa projektien alussa, sillä monet muut automaatioprojektin suunnitteluvaiheet nojautuvat I/O-listaan.

Opinnäytetyönä tehtävän työkalun avulla on tarkoitus saada projekteihin valmis I/O-lista, kun tiedetään käytettävät logiikat ja I/O-kortit sekä mahdollisesti käytettävät kenttälaitteet ja niiden toiminnot. Nykyään I/O-listat laaditaan manuaalisesti käsin, mikä vie suunnittelijalta paljon aikaa ja on samalla altis inhimillisille virheille. Työkalun tavoite on tehostaa laitteiden massaluontia ja muistuttaa suunnittelijaa mahdollisista I/O-liitynnöistä sekä yhdenmukaistaa yrityksen suunnittelua niin paljon kuin se projektien puitteissa on mahdollista.

Työkalu tehdään Visual Basic for Applications -ohjelmointikielellä, jonka avulla voidaan laatia halutun toimintasarjan suorittava ohjelma eli makro. Toimintasarja on mahdollista laatia joko nauhoittamalla Excelissä tehty toiminta tai ohjelmoimalla toimintasarja itse. VBA on Microsoft Officen ohjelmissa käytettävä ohjelmointikieli, jonka käyttöaste on suurin nimenomaan Excelissä.

Työ tehdään Insta Automation Oy:lle, jossa työskentelee satoja automaatioalan ammattilaisia. Isossa yrityksessä suunnittelun nopeuttamisella on mahdollista laskea kuluja merkittävästi, joten valmiista opinnäytetyöstä on varmasti hyötyä yrityksen tuloksen kannalta.

2 INSTA OY

Insta Oy on suomalainen riippumaton teknologiakonserni, joka toimittaa ja ylläpitää luotettavaa ja nykyaikaista automaatio-, puolustus- ja turvallisuusteknologiaa vaativiinkin olosuhteisiin. Installa on kaikkiaan 16 toimipistettä, joista 15 sijaitsee Suomessa, pääpaikan ollessa Tampereella Sarankulmassa. Vuonna 2018 työntekijöitä oli Insta Oy:llä noin 1000 ja yrityksen liikevaihto oli noin 122 miljoonaa euroa.

Insta Group konserniin kuuluu kaikkiaan viisi toimialayhtiötä:

- Insta Automation Oy
- Insta DefSec Oy
- Insta ILS Oy
- Intopalo Digital Oy
- Insta Response Oy

(Insta Group Oy 2019)

Insta Automation Oy on sähköautomaation elinkaarikumppani, jonka toiminta perustuu kattavaan asiakasymmärrykseen ja sen pohjalta rakennettuihin palveluihin. Insta Automationin vahvin kokemus on teollisuusautomaatiossa, jonka parissa Insta Automation palvelee asiakkaitaan kokonaiskumppanina, mikä tarkoittaa, että palvelut tuotetaan kokonaisratkaisuina tai erillistoimituksina asiakkaan tarpeiden mukaan. Yrityksen toimintaan kuuluvat suunnittelu, keskusvalmistus, asennuspalvelut, ylläpitopalvelut, sähkönjakelupalvelut sekä älykkään kunnossapidon ja prosessiohjauksen ratkaisut.

Insta Automationin toiminta pohjautuu Life Cycle Partner -ajatteluun eli asiakkaan sähköautomaation koko elinkaaren hallintaan ja sen avulla tuottamaan lisäarvoon. Insta Automationin osuus konsernin liikevaihdosta vuonna 2018 oli noin 60 % eli selvästi suurin osa koko konsernin liikevaihdosta. (Insta Group Oy 2019)

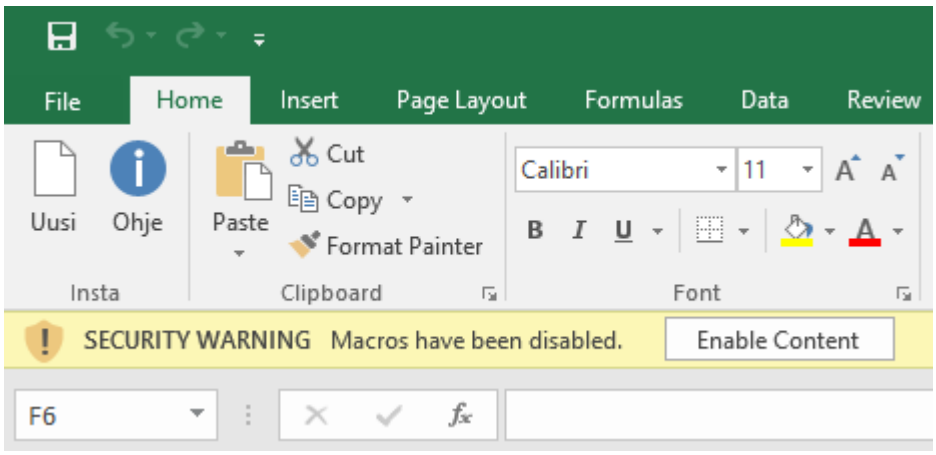
3 VBA-OHJELMOINTI

Opinnäytetyönä Insta Automation Oy:lle tehty työkalu sähkösuunnittelun laitemäärittelyn tehostamiseksi toteutettiin Visual Basic for Applications -ohjelmointikielellä. VBA-ohjelmointikieleen ja sen toimintaan perehdytään tässä luvussa siinä määrin kuin se ohjelman toimintojen ymmärtämisen kannalta on tarpeellista. Luvussa käydään läpi VBA-ohjelmoinnin ominaisuuksia ja käyttökohteita yleisesti.

3.1 VBA-kieli yleisesti

VBA-ohjelmointi on yksinkertaisimmillaan Microsoft Excelissä tehdyn toiminnan nauhoittamista toimintasarjaksi. Tällöin yhdellä napin painalluksella saadaan suoritettua haluttu toiminta. Tällaista nauhoitettua toimintasarjaa kutsutaan makrokieleksi, joka Excelissä tunnetaan nimellä Visual Basic for Applications eli VBA. VBA antaa käyttäjälle paljon mahdollisuuksia, sillä kaikki, mitä Excelissä voidaan tehdä, on tehtävissä VBA:n avulla, mutta VBA:lla voidaan suorittaa myös paljon toimintoja, joita pelkällä Excelillä ei saada suoritettua. VBA-kieli on käytettävissä myös kaikissa muissa Officen ohjelmissa, joten kieli ei rajoitu pelkästään Exceliin. Muissa Microsoft Officen ohjelmissa kuten Wordissa ja Power Pointissa VBA-kielen käyttökohteita on tosin selkeästi vähemmän. (Vaihekoski 2004, 53.)

Makrojen kautta voidaan vaikuttaa lähes kaikkeen tietokoneessa, minkä myötä makroja käytetään toisinaan haittaohjelmien levittämiseen. Microsoft on lisännyt tästä syystä Exceliin varoituksen aina, kun makroja sisältävä tiedosto avataan. Varoitusikkunasta on mahdollisuus valita, halutaanko aktivoida tiedoston makrot vai ei. Makrot kannattaa aktivoida vain, mikäli ollaan täysin varmoja, ettei tiedosto sisällä virusta. Jos ei olla varmoja tiedoston sisällöstä, kannattaa makrot jättää aktivoimatta. Tällöin Excel lataa makrot muistiin, mutta ei suorita niitä. Näin ollen VBA-kieltä tuntevilla on mahdollisuus käydä tarkistamassa makrokoodi mahdolliselta haittaohjelmaan viittaavalta sisällöltä. Varoitus on esitetty kuvassa 1. (Vaihekoski 2004, 59.)

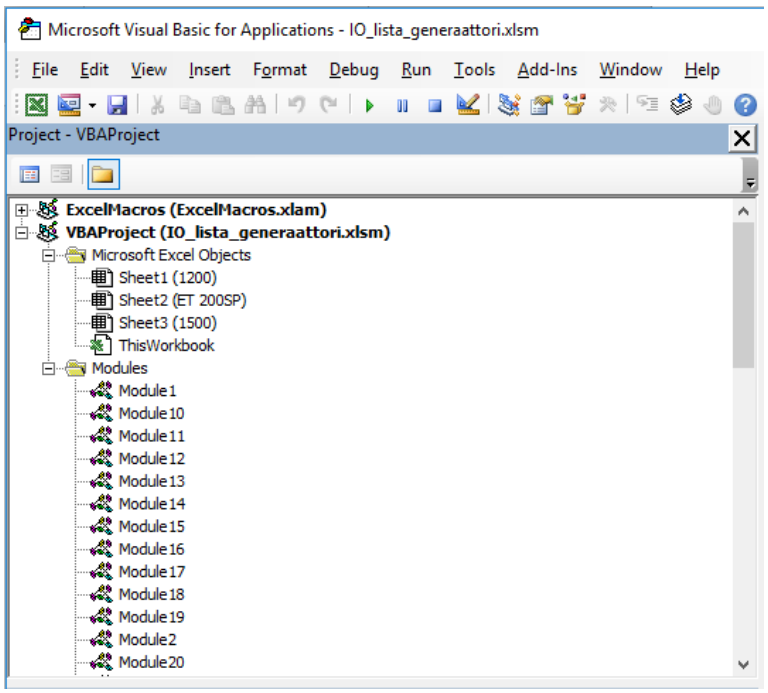


KUVA 1. Varoitus makroja sisältävästä tiedostosta.

3.2 Ohjelmoinnin aloittaminen

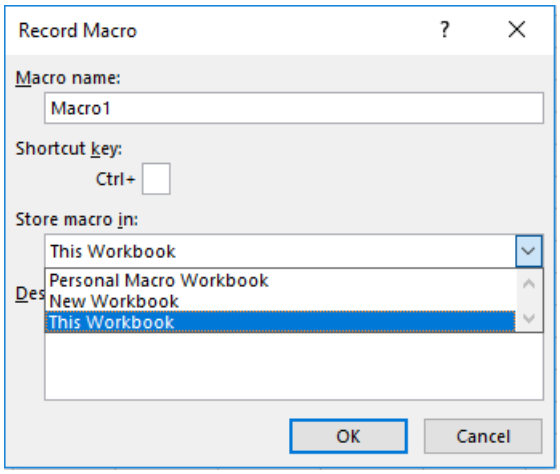
Excel voidaan siis nauhoitusominaisuudella määrätä kirjoittamaan ohjelmaa sen mukaan, mitä Excelissä tehdään, tai sitten VBA-makro voidaan kirjoittaa itse. VBA-makroja voidaan käsitellä Visual Basic Editor (VBE) -ohjelman avulla. VBE on erillinen ohjelmaikkuna, mutta on silti kiinteä Excel-ohjelman osa.

Ohjelmaikkunan vasemmasta reunasta löytyy projektihallintaikkuna, jossa näkyy avointen työkirjojen VBA-projektit. VBA-projekti voidaan tulkita ikään kuin valtaväyläksi, jossa työkirja- ja taulukko-objektit ovat sivuteitä. Ensimmäinen VBA-projektista haarautuva sivutie on Microsoft Excel Objects. Siitä haarautuu työkirjaobjekteja, jotka sisältävät taulukoita. Rakenteeseen on myös mahdollista lisätä objekteja, jotka sisältävät moduuleita. Näihin moduuleihin puolestaan on mahdollista ohjelmoida toimintasarjoja eli makroja. Makroja voidaan toki kirjoittaa myös suoraan työkirjaobjektin taulukoihinkin, mutta se ei ole välttämättä järkevää makrojen lukumäärän kasvaessa. Projektihallintaikkuna on esitetty kuvassa 2. (Shepherd 2006, 4–5.)



KUVA 2. Projektihallintaikkuna

Nauhoitettu tai kirjoitettu makro on mahdollista tallentaa johonkin kolmesta eri tallennuspaikasta. Ensimmäinen niistä on käyttäjän makrotiedosto (Personal Macro Workbook), johon tallentamalla makro on kaikissa taulukoissa käytettävissä eikä sitä ole rajoitettu mihinkään. Tämä on hyvä vaihtoehto, mikäli makro on niin sanotusti yleiskäyttöinen ja sitä käytetään jatkuvasti. Toinen tallennusmahdollisuus on uusi tiedosto (New Workbook), jolloin makro lisätään täysin uuteen tiedostoon. Kolmas vaihtoehto on tallentaa makro vain käytössä olevaan tiedostoon (This Workbook). Tällöin makro on ajettavissa vain käytössä olevassa taulukossa. Tallennuspaikan valinta on esitetty kuvassa 3. (Vaihekoski 2004, 54.)



KUVA 3. Makron tallennuspaikan valinta

Itsekirjoitettuja makroja on myös mahdollista kommentoida, jolloin niiden ymmärtäminen on helpompaa makroeditor-ohjelmassa. Kommentit alkavat aina heittomerkillä, jolloin Excel tulkitsee kaiken heittomerkin jälkeen kyseisellä rivillä kirjoitetun tekstin kommentiksi eikä täten huomioi sitä makroa ajaessa. Kommentit esitetään makroeditorissa aina vihreällä, jotta ne erottuisivat käyttäjälle paremmin. Kommenttien lisääminen makroihin onkin suotavaa, sillä se helpottaa ohjelman lukemista jälkeenpäin varsinkin, jos ohjelmaa tarkastelee jokin muu kuin ohjelman kirjoittanut henkilö. (Vaihekoski 2004, 59.)

3.3 Muuttujat ja niiden nimeäminen

VBA-ohjelmoinnissa voi tulla tilanteita, jolloin jotakin tietoa täytyy säilyttää väliaikaisesti tallessa ohjelman suorituksen aikana. Tästä syystä on tärkeä ymmärtää muuttujia koskevat säännöt.

Muuttujien määrittelyä voidaan verrata esimerkiksi kotona tärkeiden asiakirjojen arkistointiin. Eri asiakirjoille, kuten vakuutuksille tai työpapereille on luultavasti eri kansio. Näiden kansioiden koko voi vaihdella hyvinkin paljon, mutta oleellista on, että jokainen kansio on tarkoitettu vain tiettyjä asiakirjoja varten. VBA-ohjelmoinnissa nämä kansiot ovat muuttujia. Ohjelman on säilytettävä muuttuja muistissa ja viitattava siihen suorituksen aikana. Muuttujan arvo voi olla useamman merkin mittainen jono tai pelkästään yksittäinen numero. Ohjelman on mahdollista vaihtaa muuttujan arvoa muuttujan tietotyypin sallimissa rajoissa. Esimerkiksi, jos muuttujan tietotyypiksi on määritelty kokonaisluku, ei siihen voida kirjoittaa tekstiä. Jos taas kokonaislukutyyppiseen muuttujaan syötetään liukuluku, hukataan desimaaliosan sisältö. Muuttuja määritetään aina käyttämällä Dim-sanaa, jonka perään kirjoitetaan muuttujan nimi ja tietotyyppi. Esimerkiksi:

```
Dim Kokonaisluku As Integer
```

Muuttujien nimeämisessä on neljä kultaista sääntöä:

- Muuttujan nimen täytyy alkaa aina kirjaimella.
- Muuttujan nimi ei saa sisältää välilyöntejä.
    - Mikäli muuttujan nimi on pitkä ja se halutaan selkeämpilukuiseksi, voidaan välilyönti korvata alaviivalla.
- Muuttujan nimi voi olla maksimissaan 255 merkin pituinen.
- Muuttujan nimi ei saa olla varattu sana.
    - Varatut sanat ovat VBA-kieleen valmiiksi määriteltyjä avainsanoja. Mikäli näitä avainsanoja olisi mahdollista käyttää muuttujan nimenä, se voisi johtaa pahoihin sekaannuksiin ja ristiriitoihin.

Määritettäessä muuttuja toimintosarjaa varten, sitä on mahdollista käyttää vain sen toimintasarjan koodissa, jolloin muuttuja on paikallinen. Monesti kuitenkin tulee tilanne, jossa samaa muuttujaa halutaan käyttää kaikissa moduulin toimintasarjoissa. Tällöin luodaan globaali muuttuja, joka säilyttää arvonsa ohjelman käytön aikana ja jota kaikki moduulin toimintasarjat voivat käyttää. Globaali muuttuja määritellään Global-lauseella esimerkin mukaisesti. (Shepherd 2006, 14–15.)

Global Kokonaisluku

3.4 VBA:n tietotyypit

Ohjelmoitaessa VBA:lla on mahdollista käyttää useaa eri tietotyyppiä. Eri tietotyyppejä käytetään eri tilanteissa riippuen muuttujan arvoalueesta ja tyypistä. Lähtökohtaisesti muuttuja on joko kokonaisluku, liukuluku tai merkkijono. Seuraavassa luettelossa on esitetty VBA-ohjelmointikielen eri tietotyypit.

- Integer, kahden tavun kokonaisluku
- Long, neljän tavun kokonaisluku
- Single, neljän tavun liukuluku
- Double, kahdeksan tavun liukuluku
- Currency, kahdeksan tavun liukuluku, jossa kiinteä määrä desimaaleja
- String, käyttäjän määrittämä merkkijono, jossa vain tekstiä
- Variant, voi sisältää kaikkia yllä mainittuja tietotyyppejä

Pelkästään kokonaislukuja käsiteltäessä muuttujaksi kannattaa määrittää Integer tai Long lukujen koosta riippuen. Tällöin lukujen käsittely on nopeampaa ja muistia kuluu mahdollisimman vähän. (Shepherd 2006, 19.)

Mikäli ohjelmassa käsitellään desimaalilukuja, on viisainta valita tietotyypiksi Single, Double tai Currency. Currency-tietotyypin kokonaisosassa voi olla enintään 15 numeroa ja se sisältää aina neljä desimaalia. Single- ja Double-tietotyypeissä on Currencya laajempi lukualue, mutta niiden käyttö voi johtaa pieniin pyöristysvirheisiin. (Shepherd 2006, 19.)

Mikäli muuttuja sisältää tekstiä, on tietotyypiksi hyvä määrittää String. Tällöin tekstifunktiota voidaan käyttää käsiteltäessä muuttujan arvoja eroteltaessa osia merkkijonosta tai etsittäessä siitä tiettyjä merkkejä. (Shepherd 2006, 19–20.)

Kuten listasta käy ilmi, Variant-tietotyyppi sopii kaikkiin tilanteisiin, joten miksi ei käytettäisi sitä aina? Syy on hyvin yksinkertainen. Variant ei välttämättä aina osaa valita kuhunkin tilanteeseen parhainta tietotyyppiä, jolloin ohjelman suorittamiseen kuluu enemmän aikaa ja saatetaan käyttää turhaan tietokoneen muistia, jota voitaisiin käyttää muihin tarkoituksiin. Esimerkiksi tehtäessä useita yksinkertaisia laskutoimituksia pienillä kokonaisluvuilla voitaisiin käyttää Integer-tietotyyppiä, mutta Variant-tietotyyppi saattaa tulkita luvut liukulukuna ja käyttää näin ollen suotta liikaa muistia. Mikäli tietotyyppi jätetään määrittelemättä, muuttuja asettuu automaattisesti Variant-tyyppiseksi. (Shepherd 2006, 19.)

VBA:n tietotyyppejä yhdistelemällä voidaan luoda muuttujille omia tietotyyppejä. Tällöin käytetään avainsanaa Type. Esimerkiksi jos halutaan tietotyyppi, jossa esitetään palkansaajan nimi, ikä sekä palkka kirjoitetaan se seuraavasti:

```
Type Palkansaaja
        Nimi As String
        Ika As Integer
        Palkka As Currency
End Type
```

Itse luotua tietotyyppiä voidaan käyttää kuten VBA:n sisäisiä tietotyyppejä. Itse luotu tietotyyppi sisältää nyt tiedot palkansaajan nimestä, iästä ja palkasta. (Shepherd 2006, 24.)

3.5 VBA-ohjelmoinnin perusrakenteet

VBA-ohjelman on monesti tehtävä jonkinlaisia päätöksiä vastaanotetun datan perusteella. Tällaisilla päätöksillä ohjelma määrittelee, miten eri tilanteissa toimitaan, joten näiden päätöksentekorakenteiden tekeminen on hyvin keskeinen asia ohjelmoinnissa. (Shepherd 2006, 37.)

3.5.1 Ehtorakenteet

Tyypillinen ohjelman tekemä päätös noudattaa seuraavaa ehtorakenneprotokollaa: Jos (IF) asia on tosi, niin (THEN) suorita toiminto 1, muutoin (ELSE) suorita toiminto 2. Käytännössä koodi rakentuu seuraavan esimerkin mukaisesti.

```
Sub Ehtorakenne()
        If ActiveSheet.ActiveCell = 1 Then
            MsgBox ”Pidä tauko”
        Else
            MsgBox ”Jatka vielä hetki”
        End If
End Sub
```

Esimerkissä käytetty ActiveSheet.ActiveCell ilmaus viittaa Excelin aktiivisen taulukon aktiiviseen soluun. MsgBox komennolla saadaan luotua koodin ehtojen mukaisesti viestilaatikko. Mikäli aktiivisen solun arvo on yksi, ohjelma ehdottaa viestilaatikossa käyttäjää pitämään tauon. Muussa tapauksessa viestilaatikkoon tulee kehotus jatkaa vielä hetki. (Shepherd 2006, 38.)

Ehtolausetta on mahdollista myös jatkaa niin, että vaihtoehtoja tuloksen palauttamiselle on useampia. Tällöin ehtolauseessa käytetään myös komentoa ElseIf.

```
Sub Ehtorakenne()
        If ActiveSheet.ActiveCell < 2 Then
            MsgBox ”Pidä tauko”
        ElseIf ActiveSheet.ActiveCell < 4 Then
            MsgBox ”Jatka vielä hetki”
        Else
            MsgBox ”Aika lähteä kotiin”
        End If
End Sub
```

Ohjelmoijan täytyy kuitenkin olla tarkkana käyttäessään tämän tyyppistä ohjelmarakennetta. Ehtojen järjestys täytyy olla oikeanlainen, jotta ehdot poissulkevat toisensa. Mikäli ”Jatka vielä hetki” ja ”Pidä tauko” -ehtojen paikkaa vaihdettaisiin niin, että ensimmäisenä verrattaisiin aktiivisen solun dataa arvoon 4, jäisi seuraava ehtolause aina käyttämättä. Kaikki luvut jotka ovat arvoltaan alle 2 ovat myös alle 4, joten ”pienempi kuin 2” -ehtolauseeseen ei päädyttäisi ikinä. (VBA If Statement, 2020)

If-ehtolauseessa tarkastellaan tilannetta monesti niin, että mikäli jokin ehto toteutuu, suoritetaan jokin toiminto ja mikäli ehto ei toteudu, tehdään jotakin muuta. Tällaisia esimerkkejä nähtiin aikaisemmin. Ehtolauseen voi kuitenkin rakentaa myös toisinpäin, jolloin jokin toiminto suoritetaan ehdon ollessa epätosi. Tämän voi toteuttaa kahdella eri tavalla. Ensimmäisessä tavassa If-lauseeseen lisätään sana Not, jolloin ehto kääntyy päinvastaiseksi.

```
Sub Ehtorakenne()
        If Not ActiveSheet.ActiveCell = 1 Then
            MsgBox ”Pidä tauko”
        Else
            MsgBox ”Jatka vielä hetki”
        End If
End Sub
```

Toinen tapa on vaihtaa ”normaalin” If-lauseen yhtäsuuruusmerkki erisuureksi, jolloin ehto kääntyy päinvastaiseksi. Excel VBA:ssa erisuuruus toteutetaan ”pienempi kuin” ja ”suurempi kuin” -merkkien avulla. (VBA Comparison Operators, 2020)

```
Sub Ehtorakenne()
        If ActiveSheet.ActiveCell <> 1 Then
            MsgBox ”Pidä tauko”
        Else
            MsgBox ”Jatka vielä hetki”
        End If
End Sub
```

Pitkä If-ehtolause on mahdollista korvata Select Case -lauseella. Käytännössä rakenne on lähes samanlainen kuin If-lauseessa, mutta koodi on hieman selkeämpi. Seuraavassa esimerkissä on kehitetty aikaisempaa If-ehtorakennetta.

```
Sub SelectCaseRakenne(Luku)
        Select Case Luku
        Case is < 5
            MsgBox ”Pidä tauko”
        Case 5 to 10
            MsgBox ”Jatka vielä hetki”
        Case is > 10
            MsgBox ”Aika lähteä kotiin”
        Case Else
            MsgBox ”Jokin meni hassusti”
        End Select
End Sub
```

Kuten esimerkistä nähdään, Select Case -rakenteessa voidaan lukujen kanssa käyttää sanoja ”to” ja ”is”. Tämä säästää koodia useiden rivien verran, varsinkin pitkissä ehtolauseissa. (Merensalmi 2007, 83.)

3.5.2 Silmukkarakenteet

Silmukkarakenteiden avulla ohjelmoinnista saadaan helpommin hallittavaa. Silmukkarakenteen avulla koodin jotakin osaa voidaan toistaa niin kauan, kunnes jokin ehto toteutuu tai jokin annettu arvo löytyy. Tällainen toiminta saadaan aikaiseksi esimerkiksi Do Until…Loop -silmukalla:

```
Sub Silmukkarakenne()
    x = 1
        Do Until x = 10
            Cells(x, 1).Value = “Silmukka”
            x = x + 1
        Loop
End Sub
```

Alussa muuttujan arvoksi asetetaan yksi, minkä jälkeen annetaan ehto silmukan jatkamiseksi, kunnes muuttujan arvo on kymmenen. Muuttujan arvoa kasvatetaan aina yhdellä, jolloin teksti ”Silmukka” kirjoitetaan taulukon kymmenelle ensimmäiselle riville sarakkeeseen yksi. (Shepherd 2006, 43–44.)

4 AUTOMAATIOSUUNNITTELU

Automaatiojärjestelmä koostu fyysisesti mitta-antureista, toimilaitteista, ohjainlaitteista, operointilaitteista ja tiedonsiirtolaitteista. Ohjelmallisesti automaatiojärjestelmä puolestaan koostuu automaatiosovelluksista, jotka lukevat mitta-antureilta saatuja tietoja ja ohjaavat laitteita, joiden avulla säädetään prosessia. Mitta-antureilta tulevat tiedot viedään automaatiojärjestelmään tulokorttien kautta ja ohjaaville laitteille menevät ohjaustiedot viedään järjestelmästä toimilaitteille lähtökorttien avulla. Näitä tulo- ja lähtökortteja kutsutaan I/O-korteiksi (input/output). (Oulun ammattikorkeakoulu)

Automaatiojärjestelmä voi olla keskitetty tai hajautettu. Keskitetyssä automaatiojärjestelmässä yksittäinen ohjelmoitava logiikka ohjaa siihen liitettyihin I/O-kortteihin kytkettyjä laitteita. Joissakin tapauksissa myös CPU:t voivat kuitenkin itsessään sisältää I/O-kytkentäpisteitä. Tällöin erillisiä I/O-kortteja ei välttämättä tarvita, mikäli logiikkalaitteessa olevat I/O-määrät riittävät. Tällaisia logiikkalaitteita sisältää muun muassa Siemensin 1200-sarja, jonka kytkentöihin palataan myöhemmin tässä raportissa. Hajautetussa automaatiojärjestelmässä prosessin eri toiminnot on jaettu omille logiikkalaitteille. Tätä käytetään yleensä suuriin tai keskisuuriin järjestelmiin, koska se lisää koko järjestelmän luotettavuutta. Yhden logiikkalaitteen rikkoutuminen ei vaikuta koko järjestelmään vaan ainoastaan sitä ohjaavaan prosessiin. (Strömman, Hirvonen, Hukki, Tommila 2017, 10 – 12.)

Logiikkalaitteet ja I/O-kortit sijaitsevat automaatiokeskuksissa, joihin lisätään lähes poikkeuksetta ristikytkentärimat. Ristikytkennän tarkoitus on helpottaa kenttälaitteiden kytkemistä järjestelmään sekä tehdä automaatiokeskuksista siistimpiä ja turvallisempia. Automaatiokeskukset pyritään yleensä sijoittamaan mahdollisimman lähelle prosessia, jotta kaapelivedot kenttälaitteille saataisiin mahdollisimman lyhyiksi. Kuvassa 4 on esitetty automaatiokeskus, missä on nähtävissä Siemensin 1200-sarjan ohjelmoitava logiikka I/O-kortteineen (kohta 1) sekä I/O-liityntöihin kytketyt ristikytkentärimat (kohta 2). Näiden lisäksi keskuksessa on radiomodeemi sekä virransyötön suojakomponentteja kuten sulakkeita ja johdonsuojakatkaisijoita.

KUVA 4. Automaatiokeskus

Sähkösuunnittelun kannalta I/O-listan laatiminen on erittäin tärkeää, sillä monissa automaatiosuunnittelun vaiheissa tarvitaan I/O-listaan määritettyjä tietoja. I/O-listasta nähdään logiikkalaitteiden ja I/O-korttien tyyppi, osoite, kytkentäpisteet, ristikytkentäryhmä ja niiden kytkentäpisteet. I/O-listasta nähdään myös järjestelmään kytketyt kenttälaitteet sekä niiden positiot ja PLC-tagit.

Jokaiselle kenttälaitteella on aina oma positiotunnus, jonka avulla kenttälaite yksilöidään. Positiotunnuksesta käy myös ilmi, mitä mitataan tai ohjataan. PLC-tag puolestaan on kenttälaitteen muuttuja, joka sitä osoitetta vastaavaan operaatioon ohjelmassa on määritetty. Yhdellä kenttälaitteella voi olla useampia muuttujia riippuen kenttälaitteen toiminnoista. Esimerkiksi PLC-tag venttiilin V-123 ohjaus auki -komennolle olisi V-123_COPN. PLC-tagista nähdään kenttälaitteen positiotunnus sekä muuttujan suorittava toiminta (control open).

5 TYÖKALU I/O-LISTOJEN LUOMISEEN

Itse työn tekeminen alkoi tutustumalla yrityksen suunnittelijoiden aikaisempaan käytäntöön luoda I/O-listoja. Yrityksen verkkolevyltä löytyi eri projekteihin tehtyjä I/O-listoja, joita tarkasteltaessa kävi nopeasti ilmi, että manuaalisesti laadittuja I/O-listoja on käytännössä kolmea eri tyyppistä. Näistä valittiin yksi, jonka pohjalle työkalua alettiin ohjelmoida. Syy valintaan oli yksinkertainen, sillä kyseisen malliseen I/O-listaan oli tehty aikaisemmin kehitystyö piirikaavioiden suunnitteluun liittyen ja tämän haluttiin toimivan myös jatkossa. I/O-lista toimii myös testausdokumenttipohjana I/O-testejä varten sekä mahdollistaa asentajan tekemään keskuksen sisäiset kytkennät logiikoiden osalta. Kuvassa 5 on esitetty I/O-lista kokonaisuudessaan.

									KORTTI				RISTIKYTKENTÄ						
Rivi	Positio	Tarkenne	Symboli / PLC Tag	Kommentti	I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-	Piirustusnumero	Huomautus	Rev pvm	Rev kommentti
1	Z1		Z1_ALRM	Z1 Ylijännitesuoja hälytys	DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1				
2	SK1		SK1_ALRM	SK1 Sähkökatko hälytys	DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2				
3	G-123		G-123_ALRM	G-123 Veistorni ovikytkin hälytys	DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3				
4	G-123		G-123_ACKWL	G-123 Veistorni ovikytkin ohitus	DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4				
5	FIC-123		FIC-123_P	FIC-123 Tulovesi pulssi	DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5				
6	FIC-123		FIC-123_DIR	FIC-123 Tulovesi suunta	DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6				
7	M-123		M-123_RUN	M-123 Moottori 1 käy	DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7				
8	M-123		M-123_AUTO	M-123 Moottori 1 automaatilla	DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8				
9	M-123		M-123_FLT	M-123 Moottori 1 häiriö	DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9				
10	M-123		M-123_SSW	M-123 Moottori 1 turvakytkin	DI	%I1.1			A1.0		DI b.1		DIX1.0	+	10				
11	V-123		V-123_OPN	V-123 Venttiili 1 auki	DI	%I2.0			A1.1		DI a.0		DIX1.1	+	1				
12	V-123		V-123_CLS	V-123 Venttiili 1 kiinni	DI	%I2.1			A1.1		DI a.1		DIX1.1	+	2				
13	V-123		V-123_FLT	V-123 Venttiili 1 häiriö	DI	%I2.2			A1.1		DI a.2		DIX1.1	+	3				
14	V-123		V-123_TRQ	V-123 Venttiili 1 momentti	DI	%I2.3			A1.1		DI a.3		DIX1.1	+	4				
15	P-123		P-123_RUN	P-123 Pumppu 1 käy	DI	%I2.4			A1.1		DI a.4		DIX1.1	+	5				
16	P-123		P-123_FLT	P-123 Pumppu 1 häiriö	DI	%I2.5			A1.1		DI a.5		DIX1.1	+	6				
17					DI	%I2.6			A1.1		DI a.6		DIX1.1	+	7				
18					DI	%I2.7			A1.1		DI a.7		DIX1.1	+	8				
19	G-123		G-123_CTRL	G-123 Veistorni ovikytkin ilmaisin	DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-				
20	M-123		M-123_CRUN	M-123 Moottori 1 ohjaus	DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-				
21	V-123		V-123_COPN	V-123 Venttiili 1 ohjaus auki	DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-				
22	V-123		V-123_CCLS	V-123 Venttiili 1 ohjaus kiinni	DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-				
23	V-123		V-123_CSTOP	V-123 Venttiili 1 seis	DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-				
24	P-123		P-123_CTRL	P-123 Pumppu 1 ohjaus	DO	%Q0.5			A1.0	DQ b.1			DOX1.0	6	-				
25					DO	%Q1.0			A1.2	DQ a.0			DOX1.2	1	-				
26					DO	%Q1.1			A1.2	DQ a.1			DOX1.2	2	-				
27					DO	%Q1.2			A1.2	DQ a.2			DOX1.2	3	-				
28					DO	%Q1.3			A1.2	DQ a.3			DOX1.2	4	-				
29					DO	%Q1.4			A1.2	DQ a.4			DOX1.2	5	-				
30					DO	%Q1.5			A1.2	DQ a.5			DOX1.2	6	-				
31					DO	%Q1.6			A1.2	DQ a.6			DOX1.2	7	-				
32					DO	%Q1.7			A1.2	DQ a.7			DOX1.2	8	-				
33	FIC-123		FIC-123_FLW	FIC-123 Tulovesi virtaus	AI	%IW100			A1.3	0+	0-		AIX1.3	1	2				
34	M-123		M-123_FBK	M-123 Moottori 1 taajuus	AI	%IW102			A1.3	1+	1-		AIX1.3	3	4				
35					AI	%IW104			A1.3	2+	2-		AIX1.3	5	6				
36					AI	%IW106			A1.3	3+	3-		AIX1.3	7	8				
37	M-123		M-123_CRPM	M-123 Moottori 1 ohje	AO	%QW100			A1.0	0	3M		AOX1.0	1	2				
38					AO	%QW102			A1.0	1	3M		AOX1.0	3	4				
39					AO	%QW104			A1.4	0	0M		AOX1.4	1	2				
40					AO	%QW106			A1.4	1	1M		AOX1.4	3	4				

KUVA 5. Yleiskuva I/O-listasta

5.1 Työkalun toiminnot

I/O-listatyökalun tarkoitus on avustaa HW-suunnittelijaa määrittämään käytetyimpien logiikkajärjestelmien I/O-listat. I/O-listasta käy ilmi käytettyjen logiikkalaitteiden ja I/O-korttien tyyppi, osoite, kytkentäpisteet ja ristikytkentäryhmä sekä niiden

kytkentäpisteet. Nämä kaikki vaihtelevat I/O-tyypin mukaan. Esimerkiksi digitaalisella tulotiedolla on erityyppinen osoite kuin analogisella lähtötiedolla.

Eri logiikkaperheiden I/O-listapohjien lisäksi työkaluun lisättiin kattava kirjasto yrityksen toimialueen käytetyimmistä kenttälaitteista kuten esimerkiksi moottoreista ja venttileistä. Kirjasto sisältää VBA:lla ohjelmoitua koodia, minkä avulla I/O-listaan halutut kenttälaitteet lisätään listaan automaattisesti. Kirjaston tavoite on muistuttaa suunnittelijaa mahdollisista I/O-pisteistä ja näin ollen vähentää inhimillisten virheiden määrää. Kirjaston avulla suunnittelijan on helppo valita haluamansa kenttälaite ja lisätä se I/O-listaan. Lisättäessä kenttälaite I/O-listaan, kirjoitetaan siihen laitteen nimi, PLC-tag sekä positio. Näiden parametrien kirjoittaminen suoraan kirjaston avulla yhdenmukaistaa suunnittelua ja näin ollen yksinkertaistaa I/O-listojen ymmärtämistä sekä helpottaa suunnittelijoiden ja asentajien töitä myöhemmässä vaiheessa.

5.2 I/O-listan ohjelmointi

Aikaisemmin tehtyihin I/O-listoihin perehdyttäessä taulukon perusajatus kävi nopeasti selväksi. Listassa esitetään ensimmäisenä digitaaliset tulot, sitten digitaaliset lähdöt, jonka jälkeen esitetään analogiset tulot ja viimeisenä analogiset lähdöt. Kyseinen järjestys on hyvin yleisesti käytetty malli, joten sitä oli turha lähteä vaihtamaan.

I/O-listapohja tehtiin Siemensin ja Schneiderin logiikkaperheisiin, joista ensimmäisenä perehdyttiin tarkemmin Siemensiin, koska siitä oli eniten kokemusta entuudestaan. Kollegoiden haastattelun jälkeen oli selvää, että työkalua ruvetaan kehittämään Siemensin 1200-, 1500- sekä ET 200SP -järjestelmille. Vaikka kaikissa Siemensin järjestelmissä perusperiaate parametrien nimeämisessä on sama, laadittiin jokaisen järjestelmän logiikoille ja I/O-korteille omat VBA-makrot. Näin tehtiin myös Schneiderin kohdalla, jossa I/O-listapohja luotiin Basic-, M340- ja M580 -järjestelmille.

5.2.1 I/O-tyyppi ja kortin tunnus

I/O-kortteja voidaan lisätä I/O-listaan mielivaltaisesti niiden järjestyessä aina oikealle kohdalle tietotyypistä riippuen. Logiikkalaite puolestaan lisätään aina I/O-listaan ensimmäiseksi ja tästä syystä on suotavaa, että se lisätään ennen I/O-kortteja. Tällä tavoin logiikkalaite saadaan aina listan ensimmäiseksi ja voidaan nimetä normaalin käytännön mukaan tunnuksella A1.0. Kuvassa 6 on esitetty osa I/O-listasta. Tarkasteltava kohta on korostettu kuvaan punaisella kehyksellä.

				KORTTI				RISTIKYTKENTÄ		
I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-
DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1
DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2
DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3
DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4
DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5
DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6
DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7
DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8
DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9
DI	%I1.1			A1.0		DI b.1		DIX1.0	+	10
DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-
DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-
DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-
DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-
DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-
DO	%Q0.5			A1.0	DQ b.1			DOX1.0	6	-
AO	%QW100			A1.0	0	3M		AOX1.0	1	2
AO	%QW102			A1.0	1	3M		AOX1.0	3	4

KUVA 6. I/O-listaan lisätyn logiikkalaitteen tunnus

Kuvassa 7 on esitetty ohjelmakoodi logiikkalaitteen tunnuksen lisäämiseksi. Koodista tehtiin hyvin yksinkertainen, jotta se kuormittaisi Exceliä mahdollisimman vähän. Kuvan mukainen yksinkertainen koodi oli mahdollista toteuttaa, koska logiikkalaite lisätään I/O-listaan ensimmäisenä, eikä sen siksi tarvitse ottaa huomioon I/O-listaan mahdollisesti aikaisemmin lisättyjä I/O-kortteja.

```
' Lisätään logiikan tunnus

Dim x As Integer

For x = 1 To 18
Cells(x + 2, 10).Value = "A1.0"
Next x
```

KUVA 7. Koodi logiikkalaiteen tunnuksen lisäämiseksi I/O-listaan

Tarvittaessa lisättävien I/O-korttien tunnus nimetään juoksevalla numerolla seuraavan kortin tunnuksen ollessa A1.1. I/O-kortin tunnuksen kanssa onkin syytä olla tarkkana, sillä taulukossa tunnus ei välttämättä juoksekaan liukuvasti pienimmästä suurimpaan. Kuvassa 8 on esitetty tilanne, jossa listaan on lisätty logiikkalaitteen lisäksi yksi kortti jokaista I/O-tyyppiä. Logiikkalaite (tunnuksella A1.0) itsessään sisältää kymmenen DI-, kuusi DO- ja kaksi AO-tietoa. Tällöin lisätyn DI-kortin (A1.1) jälkeen tulevat ensimmäiset DO-tiedot ovatkin logiikkalaitteen I/O-tietoja ja näin ollen niiden tunnus on A1.0. Samanlainen tilanne esiintyy AO-tietojen kohdalla. Jälkeenpäin lisättyjen I/O-korttien tunnukset etenevät järkevästi A1.1 – A1.4.

				KORTTI				RISTIKYTKENTÄ		
I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-
DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1
DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2
DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3
DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4
DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5
DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6
DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7
DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8
DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9
DI	%I1.1			A1.0		DI b.1		DIX1.0	+	10
DI	%I2.0			A1.1		DI a.0		DIX1.1	+	1
DI	%I2.1			A1.1		DI a.1		DIX1.1	+	2
DI	%I2.2			A1.1		DI a.2		DIX1.1	+	3
DI	%I2.3			A1.1		DI a.3		DIX1.1	+	4
DI	%I2.4			A1.1		DI a.4		DIX1.1	+	5
DI	%I2.5			A1.1		DI a.5		DIX1.1	+	6
DI	%I2.6			A1.1		DI a.6		DIX1.1	+	7
DI	%I2.7			A1.1		DI a.7		DIX1.1	+	8
DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-
DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-
DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-
DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-
DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-
DO	%Q0.5			A1.0	DQ b.1			DOX1.0	6	-
DO	%Q1.0			A1.2	DQ a.0			DOX1.2	1	-
DO	%Q1.1			A1.2	DQ a.1			DOX1.2	2	-
DO	%Q1.2			A1.2	DQ a.2			DOX1.2	3	-
DO	%Q1.3			A1.2	DQ a.3			DOX1.2	4	-
DO	%Q1.4			A1.2	DQ a.4			DOX1.2	5	-
DO	%Q1.5			A1.2	DQ a.5			DOX1.2	6	-
DO	%Q1.6			A1.2	DQ a.6			DOX1.2	7	-
DO	%Q1.7			A1.2	DQ a.7			DOX1.2	8	-
AI	%IW100			A1.3	0+	0-		AIX1.3	1	2
AI	%IW102			A1.3	1+	1-		AIX1.3	3	4
AI	%IW104			A1.3	2+	2-		AIX1.3	5	6
AI	%IW106			A1.3	3+	3-		AIX1.3	7	8
AO	%QW100			A1.0	0	3M		AOX1.0	1	2
AO	%QW102			A1.0	1	3M		AOX1.0	3	4
AO	%QW104			A1.4	0	0M		AOX1.4	1	2
AO	%QW106			A1.4	1	1M		AOX1.4	3	4

KUVA 8. I/O-lista, johon lisätty logiikkalaitteen lisäksi I/O-kortteja

Tunnus määräytyy I/O-listaan oikein I/O-listan senhetkisestä tilanteesta riippumatta. Lähtökohtaisesti koodi on toteutettu If- ja For-lauseilla, joiden avulla selvitetään, onko I/O-listassa logiikkakortteja entuudestaan. Mikäli aikaisempia logiikkakortteja ei ole, tilanne on yksinkertainen ja lisätyn I/O-kortin tunnukseksi asetetaan A1.0.

Mahdollisesti I/O-listassa on logiikkalaitteen lisäksi useita muita I/O-kortteja, jolloin ohjelman täytyy ottaa ne huomioon ja osata tulkita myös niiden tietotyyppi. Tästä syystä ohjelma monimutkaistuu merkittävästi. Tällaisessa tilanteessa ohjelma lukee I/O-listan tunnussaraketta lisätyn I/O-kortin positiosta ylöspäin ja tallentaa suurimman senhetkisen tunnuksen muistiin. Kun suurin tunnus on tiedossa, kirjoitetaan lisätyn kortin tunnukseksi yhtä arvoa suurempi tunnus kuin senhetkinen suurin tunnus. Tämän suorittava ohjelmakoodi on esitetty kuvassa 9.

```
' lisätään kortin tunnus

' Mikäli solussa J3 on jotakin dataa

    If Range("J3").Value <> "" Then

    ' niin luetaan sarakkeen J suurin tunnus lisätystä kortista ylöspäin

    Dim StartRow As Integer
    Dim EndRow As Integer
    Dim i As Integer
    Dim LmaxTemp As Integer
    Dim Lmax
    Dim Temp2
    Lmax = 0
    StartRow = 3
    EndRow = Range("J3").End(xlDown).Row

    For i = StartRow To EndRow

    Temp2 = Cells(i, 10)
    LmaxTemp = Mid(CStr(Temp2), 4, 2)
    If LmaxTemp > Lmax Then
        Lmax = LmaxTemp
    End If
    Next

    ' Kirjoitetaan yhtä arvoa suurempi tunnus

    Range("L3").End(xlDown).Offset(1, -2).Select
    Do While y < 4
        ActiveCell.Value = "A1." & Lmax + 1
        ActiveCell.Offset(1, 0).Select
        y = y + 1
    Loop
```

KUVA 9. Lisätyn I/O-kortin tunnuksen määritys

Lisätyn I/O-kortin tunnuksen kirjoituksen jälkeen kasvatetaan lisätyn I/O-kortin alapuolelle jääneiden I/O-korttien kaikkia tunnuksia yhdellä, jotka ovat suurempia tai yhtä suuria kuin lisätyn I/O-kortin tunnus. Tämä suoritetaan lähes vastaavanlaisesti kuin kuvassa 9 esitetyssä koodissa.

5.2.2 Osoitteet

I/O-pisteiden logiikkaosoitteiden määrityksessä on huomioitava muutama oleellinen seikka. Kaikki osoitteet tietotyypistä riippumatta alkavat %-merkillä. Tämä siitä syystä, että SW-suunnittelussa %-merkki osoitteen alussa on välttämätön ohjelman toiminnan kannalta. Nyt, kun %-merkki on lisätty jo I/O-listassa osoitteeseen, ei SW-suunnittelijan tarvitse käyttää turhaa aikaa osoitteen uudelleenmerkitsemiseen myöhemmässä vaiheessa.

Toinen osoitteisiin liittyvä seikka on osoitteiden tietotyyppitunnisteen eroaminen toisistaan. Digitaaliset tulot merkitään tunnisteella I, digitaaliset lähdöt tunnisteella Q, analogiset tulot tunnisteella IW ja analogiset lähdöt tunnisteella QW. Nämä osoitetunnisteet määräytyvät myös SW-suunnittelun kautta, koska monien logiikkalaitevalmistajien logiikkaohjelmointi pakottaa käyttämään kyseisiä tunnisteita.

Kolmas huomion arvoinen asia on osoitteiden numeerinen kulku. Analogisten tietojen osalta asia on varsin yksinkertainen. Analogiset tulot alkavat osoitteesta %IW100. Yhden analogisen tiedon varatessa aina kaksi tavua löytyy seuraava tulo osoitteesta %IW102. Näin ollen esimerkiksi neljäkanavainen analoginen tulokortti varaa osoitteet %IW100 – %IW107. Lisättäessä tämän perään toinen AI-kortti, jatkuvat osoitteet suoraan aikaisemmasta. Esimerkkitapauksessa (kuva 10) toisen AI-kortin ensimmäisen kanavan osoite on siis %IW108. Analogisilla lähdöillä tilanne on lähes vastaava kuin analogisilla tuloilla, mutta osoite kirjoitetaan tunnisteella QW. Esimerkiksi ensimmäisen analogisen lähtötiedon osoite on siis %QW100. (Kuva 10, kohta 1)

Digitaalisten tietojen osalta asia on hieman erilainen. Jokainen digitaalinen tieto varaa yhden bitin. Digitaaliset tiedot alkavat tavun nolla bitistä nolla. Näin ollen

ensimmäinen digitaalisen tulotiedon osoite on %I0.0. Yksi tavu sisältää kahdeksan bittiä, joten viimeinen bitti ensimmäisessä tavussa on osoitteessa %I0.7. Näin ollen seuraava digitaalinen tulotieto kirjoitetaan osoitteeseen %I1.0. Digitaalisia I/O-kortteja lisättäessä on huomioitava myös, että uuden kortin osoitteet alkavat aina uudella tavulla. Havainnollistetaan tätä esimerkin avulla: Siemensin 1217C -logiikassa itsessään on kymmenen paikkaa digitaalisille tuloille. Näin ollen niiden osoitteet ovat %I0.0 – %I0.7 ja %I1.0 – %I1.1. Lisättäessä tämän perään digitaalinen tulokortti, alkaa sen tiedot seuraavasta tavusta eli osoitteesta %I2.0 (kuva X, kohta 2). Digitaalisilla lähdöillä toiminta on lähes vastaava kuin digitaalisilla tuloillakin, mutta osoite kirjoitetaan tunnisteella Q (kuva 10, kohta 3). Esimerkiksi ensimmäinen digitaalinen lähtötieto on osoitteessa %Q0.0. Osoitteet ja niiden määritys käy ilmi kuvasta 10.

				KORTTI				RISTIKYTKENTÄ		
I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-
DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1
DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2
DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3
DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4
DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5
DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6
DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7
DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8
DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9
DI	%I1.1	2		A1.0		DI b.1		DIX1.0	+	10
DI	%I2.0			A1.1		DI a.0		DIX1.1	+	1
DI	%I2.1			A1.1		DI a.1		DIX1.1	+	2
DI	%I2.2			A1.1		DI a.2		DIX1.1	+	3
DI	%I2.3			A1.1		DI a.3		DIX1.1	+	4
DI	%I2.4			A1.1		DI a.4		DIX1.1	+	5
DI	%I2.5			A1.1		DI a.5		DIX1.1	+	6
DI	%I2.6			A1.1		DI a.6		DIX1.1	+	7
DI	%I2.7			A1.1		DI a.7		DIX1.1	+	8
DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-
DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-
DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-
DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-
DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-
DO	%Q0.5	3		A1.0	DQ b.1			DOX1.0	6	-
DO	%Q1.0			A1.2	DQ a.0			DOX1.2	1	-
DO	%Q1.1			A1.2	DQ a.1			DOX1.2	2	-
DO	%Q1.2			A1.2	DQ a.2			DOX1.2	3	-
DO	%Q1.3			A1.2	DQ a.3			DOX1.2	4	-
DO	%Q1.4			A1.2	DQ a.4			DOX1.2	5	-
DO	%Q1.5			A1.2	DQ a.5			DOX1.2	6	-
DO	%Q1.6			A1.2	DQ a.6			DOX1.2	7	-
DO	%Q1.7			A1.2	DQ a.7			DOX1.2	8	-
AI	%IW100			A1.3	0+	0-		AIX1.3	1	2
AI	%IW102			A1.3	1+	1-		AIX1.3	3	4
AI	%IW104			A1.3	2+	2-		AIX1.3	5	6
AI	%IW106			A1.3	3+	3-		AIX1.3	7	8
AO	%QW100			A1.0	0	3M		AOX1.0	1	2
AO	%QW102	1		A1.0	1	3M		AOX1.0	3	4
AO	%QW104			A1.4	0	0M		AOX1.4	1	2
AO	%QW106			A1.4	1	1M		AOX1.4	3	4

KUVA 10. I/O-osoitteen määritys

Osoitetietoja lisättäessä ohjelman on otettava osoitteiden oikeellisuuden lisäksi huomioon myös oikea sijainti, mihin osoitteet kirjoitetaan. Jos esimerkiksi lisätään AI-kortti I/O-listaan, jossa ei entuudestaan ole yhtään AI-tietoja, täytyy ohjelman osata lisätä osoitteet alkavaksi viimeisen DO-osoitteen perään. Mikäli taas AI-tietoja on jo entuudestaan, täytyy lisätyn kortin osoitteet lisätä jo olemassa olevien AI-osoitteiden perään huomioiden tietenkin osoitteiden oikea jatkuvuus. Tämän suorittavasta koodista on esimerkki kuvassa 11.

```
' Mikäli löytyy AI-osoitteita

  If Not FindAI Is Nothing Then

  ' Luetaan aikaisemman AI-tiedon osoite

  Dim val As Variant
  Dim res As Variant
  Range("G3").End(xlDown).Select
  val = ActiveCell.Value
  res = Mid(val, 4, 3)

' Lisätään 4 osoitetta

  Range("G3").End(xlDown).Offset(1, 0).Select
  Do While x < 4
    Dim y As Integer
    For y = 1 To 4
    ActiveCell.Value = "%IW" & res + 2 * y
    ActiveCell.Offset(1, 0).Select
    x = x + 1
    Next y
  Loop
```

KUVA 11. AI-kortin osoitteiden lisäys

5.2.3 I/O-kortin liittimet ja ristikytkentä

I/O-korttien kytkentää varten jokaisen kortin jokainen I/O-kytkentäpiste lisättiin I/O-listaan. Korttien kytkentäpisteet vastaavat tyypiltään tarkasti valmistajan ilmoittamia, jotta väärinymmärrys asennusvaiheessa olisi mahdollisimman pieni.

Ristikytkentäryhmät nimetään ja erotellaan I/O-korttien tunnuksen ja tyypin mukaan. Ristikytkentäryhmän numerointi jäljittelee I/O-kortin tunnusta ja ristikytkennän nimestä käy selkeästi ilmi ryhmän kautta kulkevan signaalin I/O-tyyppi. Näin ollen kytkennöistä saadaan selkeitä ja siistejä, mikä puolestaan vähentää jälleen

virhekytkentöjen mahdollisuutta. Ristikytkentäryhmien nimeäminen on korostettu kuvassa 12 punaisella kehyksellä.

				KORTTI				RISTIKYTKENTÄ		
I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-
DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1
DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2
DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3
DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4
DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5
DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6
DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7
DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8
DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9
DI	%I1.1			A1.0		DI b.1		DIX1.0	+	10
DI	%I2.0			A1.1		DI a.0		DIX1.1	+	1
DI	%I2.1			A1.1		DI a.1		DIX1.1	+	2
DI	%I2.2			A1.1		DI a.2		DIX1.1	+	3
DI	%I2.3			A1.1		DI a.3		DIX1.1	+	4
DI	%I2.4			A1.1		DI a.4		DIX1.1	+	5
DI	%I2.5			A1.1		DI a.5		DIX1.1	+	6
DI	%I2.6			A1.1		DI a.6		DIX1.1	+	7
DI	%I2.7			A1.1		DI a.7		DIX1.1	+	8
DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-
DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-
DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-
DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-
DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-
DO	%Q0.5			A1.0	DQ b.1			DOX1.0	6	-
DO	%Q1.0			A1.2	DQ a.0			DOX1.2	1	-
DO	%Q1.1			A1.2	DQ a.1			DOX1.2	2	-
DO	%Q1.2			A1.2	DQ a.2			DOX1.2	3	-
DO	%Q1.3			A1.2	DQ a.3			DOX1.2	4	-
DO	%Q1.4			A1.2	DQ a.4			DOX1.2	5	-
DO	%Q1.5			A1.2	DQ a.5			DOX1.2	6	-
DO	%Q1.6			A1.2	DQ a.6			DOX1.2	7	-
DO	%Q1.7			A1.2	DQ a.7			DOX1.2	8	-
AI	%IW100			A1.3	0+	0-		AIX1.3	1	2
AI	%IW102			A1.3	1+	1-		AIX1.3	3	4
AI	%IW104			A1.3	2+	2-		AIX1.3	5	6
AI	%IW106			A1.3	3+	3-		AIX1.3	7	8
AO	%QW100			A1.0	0	3M		AOX1.0	1	2
AO	%QW102			A1.0	1	3M		AOX1.0	3	4
AO	%QW104			A1.4	0	0M		AOX1.4	1	2
AO	%QW106			A1.4	1	1M		AOX1.4	3	4

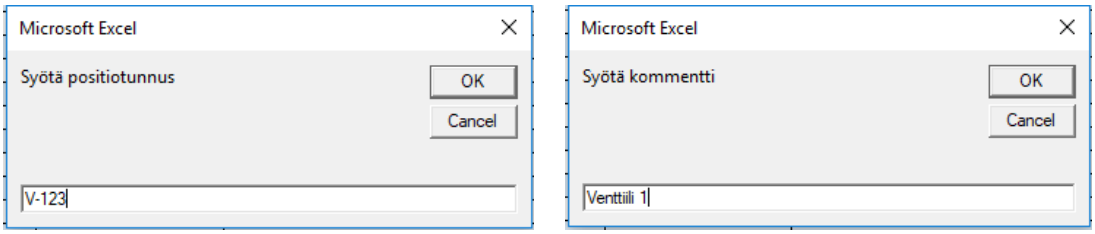
KUVA 12. Ristikytkentöjen nimeäminen

Ohjelmallisesti ristikytkentäryhmien nimeäminen noudattaa hyvin pitkälti samaa kaavaa kuin logiikkakorttien tunnusten nimeäminenkin. Ristikytkentäryhmästä nähtävä signaalin I/O-tyyppi on määritetty yksinkertaisesti jokaisen I/O-kortin ohjelmaan itseensä. Ristikytkentäryhmän numerointi, joka vastaa I/O-kortin tunnusta, suoritetaan ohjelmakoodissa lähes samalla tavalla kuin aikaisemmin esitetyssä I/O-kortin tunnuksessa.

5.3 Laitekirjaston ohjelmointi

I/O-listapohjan lisäksi työkaluun ohjelmoitiin myös yrityksen automaatioprojekteissa käytetyimmät kenttälaitteet. Tämän avulla voidaan vähentää inhimillisten virheiden määrää työkalun muistuttaessa suunnittelijaa tarvittavista I/O-pisteistä. Laaditun kirjaston avulla suunnittelijan on yksinkertaista valita kenttälaite ja lisätä se I/O-listaan. Kaikki kirjastoon luodut kenttälaitteet sekä niiden I/O-määrät ovat esitetty liitteessä 1.

Lisättäessä kirjastosta jokin kenttälaite I/O-listaan, kysytään käyttäjältä laitteen positiotunnus sekä kommentti. Valitun kenttälaitteen ja käyttäjältä kysyttyjen tietojen perusteella I/O-listaan kirjoitetaan laitteen nimi, PLC-tag sekä positio. Kuvassa 13 on esitetty tilanne, jossa suunnittelija on lisännyt I/O-listaan kenttälaitteen ja ohjelma kysyy siihen liittyen tarvittavat tiedot.



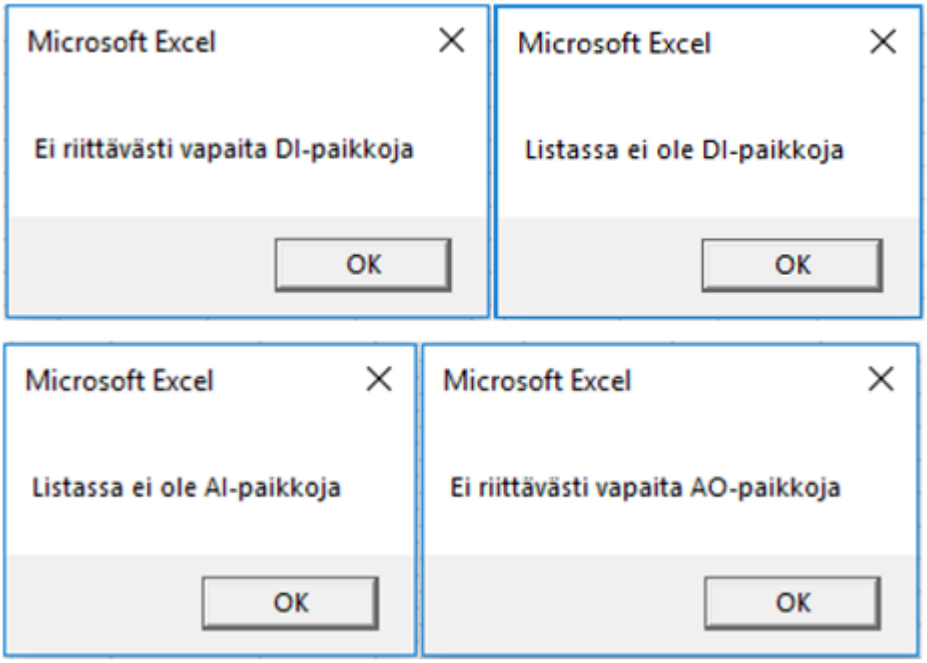
KUVA 13. Kenttälaitteen positiotunnuksen ja kommentin lisäys

Kuvassa 14 nähdään, kun I/O-listaan on lisätty muutama yleisesti käytetty kenttälaite. Listassa on murtohälytys, virtausmittaus sekä moottorin ja venttiilin ohjaukset. Automaatiokeskuksen sisäisestä toiminnasta kertovat ylijännitesuoja- ja sähkökatkohälytykset. Tarkasteltaessa kuvia 13 ja 14 huomataan, kuinka käyttäjän syöttämät positiotunnus ja kommentti näkyvät I/O-listassa. Kommenttikenttään tulevat tarkennukset kenttälaitteen toiminnasta sekä PLC-tagit ohjelma kirjoittaa automaattisesti. Huomionarvoista on, että PLC-tagissa ei voida käyttää skandinaavisia kirjaimia, sillä suunnittelun myöhemmässä vaiheessa käytettävät muut ohjelmat eivät yleensä ymmärrä niitä. Kuvassa 14 on esitetty osa I/O-listasta, josta käy ilmi lisätyt kenttälaitteet.

Rivi	Positio	Tarkenne	Symboli / PLC Tag	Kommentti	I/O-tyyppi	Osoite
1	Z1		Z1_ALRM	Z1 Ylijännitesuoja hälytys	DI	%I0.0
2	SK1		SK1_ALRM	SK1 Sähkökatko hälytys	DI	%I0.1
3	G-123		G-123_ALRM	G-123 Veistorni ovikytkin hälytys	DI	%I0.2
4	G-123		G-123_ACKWL	G-123 Veistorni ovikytkin ohitus	DI	%I0.3
5	FIC-123		FIC-123_P	FIC-123 Tulovesi pulssi	DI	%I0.4
6	FIC-123		FIC-123_DIR	FIC-123 Tulovesi suunta	DI	%I0.5
7	M-123		M-123_RUN	M-123 Moottori 1 käy	DI	%I0.6
8	M-123		M-123_AUTO	M-123 Moottori 1 automaatilla	DI	%I0.7
9	M-123		M-123_FLT	M-123 Moottori 1 häiriö	DI	%I1.0
10	M-123		M-123_SSW	M-123 Moottori 1 turvakytkin	DI	%I1.1
11	V-123		V-123_OPN	V-123 Venttiili 1 auki	DI	%I2.0
12	V-123		V-123_CLS	V-123 Venttiili 1 kiinni	DI	%I2.1
13	V-123		V-123_FLT	V-123 Venttiili 1 häiriö	DI	%I2.2
14	V-123		V-123_TRQ	V-123 Venttiili 1 momentti	DI	%I2.3
15					DI	%I2.4
16					DI	%I2.5
17					DI	%I2.6
18					DI	%I2.7
19	G-123		G-123_CTRL	G-123 Veistorni ovikytkin ilmaisin	DO	%Q0.0
20	M-123		M-123_CRUN	M-123 Moottori 1 ohjaus	DO	%Q0.1
21	V-123		V-123_COPN	V-123 Venttiili 1 ohjaus auki	DO	%Q0.2
22	V-123		V-123_CCLS	V-123 Venttiili 1 ohjaus kiinni	DO	%Q0.3
23	V-123		V-123_CSTOP	V-123 Venttiili 1 seis	DO	%Q0.4
24					DO	%Q0.5
25					DO	%Q1.0
26					DO	%Q1.1
27					DO	%Q1.2
28					DO	%Q1.3
29					DO	%Q1.4
30					DO	%Q1.5
31					DO	%Q1.6
32					DO	%Q1.7
33	FIC-123		FIC-123_FLW	FIC-123 Tulovesi virtaus	AI	%IW100
34	M-123		M-123_FBK	M-123 Moottori 1 taajuus	AI	%IW102
35					AI	%IW104
36					AI	%IW106
37	M-123		M-123_CRPM	M-123 Moottori 1 ohje	AO	%QW100
38					AO	%QW102
39					AO	%QW104
40					AO	%QW106

KUVA 14. Kenttälaitteita lisättynä I/O-listaan

Mikäli suunnittelija haluaa lisätä jonkin kenttälaitteen I/O-listaan, mutta siinä ei ole riittävästi tarvittavia I/O-pisteitä vapaana tai kyseistä I/O-tyyppiä ei ole listassa ollenkaan, ei laitetta lisätä, vaan I/O-pisteiden vajavaisuudesta annetaan suunnittelijalle ilmoitus. Osa huomautuksista on esitetty kuvassa 15.



KUVA 15. Ilmoituksia I/O-pisteiden vajavaisuuksista

Ilmoitus annetaan mahdollisimman tarkasti, ja mikäli vajavaisuuksia on useammassa kuin yhdessä paikassa, kerrotaan käyttäjälle niistä kaikista. Näin ollen varmistutaan, että I/O-listaan tulisi mahdollisimman vähän virheitä ja ettei tarvittava I/O-määrä jää ainakaan liian alhaiseksi. Kuvassa 16 on esitetty koodi, missä tarkastetaan riittävä DI-paikkojen määrä lisätylle kenttälaitteelle.

```
' Etsitään, onko DI tietoja ollenkaan

Dim FindDI As Range
Range("F3:F4").Select
Set FindDI = Selection.Find(what:="DI")

' Jos DI-paikkoja ei ole, annetaan käyttäjälle ilmoitus

If FindDI Is Nothing Then

MsgBox "Listassa ei ole DI-paikkoja"

ElseIf Not FindDI Is Nothing Then

' Tarkastetaan onko vapaita DI-paikkoja jäljellä

 Dim DIEndRow As Long
          With ActiveSheet
             DIEndRow = .Range("F:F").Find(what:="DI", After:=.Range("F3"), SearchDirection:=xlPrevious).Select
          End With
ActiveCell.Offset(-3, -1).Select

Dim VapaaDI As Variant
Set VapaaDI = Selection.Find(what:="")

    'Jos vapaita DI-paikkoja ei ole, annetaan käyttäjälle ilmoitus

If VapaaDI Is Nothing Then

MsgBox "Ei riittävästi vapaita DI-paikkoja"

End If
End If
```

KUVA 16. Koodi vapaiden DI-tietojen selvittämiseksi

6 POHDINTA

Työn tekeminen alkoi yrityksen suunnittelijoiden aikaisemmin tehtyjen I/O-listojen läpikäynnillä ja sopivan mallin valinnalla. Valinta sinänsä oli hyvin helppo, sillä kyseisen malliseen I/O-listaan oli aikaisemmin tehty kehitystyö piirikaavioiden suunnitteluun liittyen. Malli oli todettu toimivaksi ja aikaisemman kehitystyön haluttiin tietenkin toimivan myös jatkossa, joten I/O-listapohjaa ei lähdetty muuttamaan.

Työn ensimmäinen vaihe oli luoda ohjelmakirjasto Siemensin logiikkalaitteille. Kirjasto luotiin kaikkiaan kolmeen eri Siemensin logiikkaperheeseen, jotka olivat 1200-sarja, 1500-sarja sekä hajautettu ET 200SP-sarja. Kirjastojen luonti aloitettiin 1200-sarjasta, sillä siitä oli eniten aikaisempaa kokemusta ja näin ollen se todettiin helpoimmaksi toteuttaa. Lopulta eri logiikkaperheiden välille ei syntynyt merkittäviä eroja, sillä kirjaston luonnin periaate oli kaikissa logiikkaperheissä lähes samanlainen. Suurimmat erot tulivat kytkentäpisteiden nimeämisessä sekä I/O-liityntöjen lukumäärissä.

Työn aikana kommunikoitiin niin HW- kuin SW-suunnittelijoiden kanssa I/O-listaan liittyvistä asioista hyvin laajasti. Oli erittäin tärkeää, että jokainen parametri luodaan oikeaan muotoon, jotta I/O-listasta kerättävät tiedot ovat suoraan yhteensopivia muiden ohjelmien kanssa, joita käytetään suunnittelun myöhemmissä vaiheissa. Työn aikana kommunikointi kollegoiden kanssa osoittautui muutenkin tärkeäksi, koska vain heidän mielipiteidensä avulla työkalusta saatiin tehtyä kokonaisuus, joka toimii ja auttaa mahdollisimman laajalti suunnittelutyötä riippumatta työntekijän osastosta tai erikoistumisalueesta. Kommunikoinnin avulla osaston muut suunnittelijat saatiin myös tietoiseksi, että kytkentälistojen luomisen helpottamiseksi ollaan tekemässä työkalua. Lopputuloksena syntynyt työkalu on helppokäyttöinen kaikille automaatioalan insinööreille.

Valmiin työn vaikutukset eivät kuitenkaan rajoitu vain suunnittelijoiden työhön, sillä kytkentälistojen yhdenmukaistaminen helpottaa myös asentajien työskentelyä myöhemmässä vaiheessa kytkentälistojen helppolukuisuuden myötä. Myös

asennusvaiheessa tapahtuvien inhimillisten virheiden määrää saataisiin laskettua, mikäli kaikissa projekteissa käytettäisiin saman tyyppisiä I/O-listoja, eikä asentajien tarvitsisi yrittää tulkita monen eri suunnittelijan tekemiä toisistaan poikkeavia I/O-listoja.

Työkalun ohjelmallista toimivuutta testattiin jatkuvasti ohjelmoinnin edetessä, mutta työkalun toimivuutta oikeassa automaatioprojektissa ei ole testattu, koska työkalua ei ole vielä ehditty ottaa käyttöön. Ennen työkalun käyttöönottoa yrityksen suunnittelijoille pidetään lyhyt koulutus, jonka tarkoituksena on helpottaa uuden toimintamallin hyväksymistä ja vanhojen työtapojen vaihtamista uuteen.

Tulevaisuudessa työkalun jatkokehitys on mahdollista ohjelmoimalla siihen lisää eri valmistajien logiikkalaitteita sekä lisäämällä kenttälaitteita tulevien tarpeiden mukaan.

LÄHTEET

Insta Group Oy. 2019. Luettu 7.1.2020.
https://www.insta.fi/

Merensalmi, J. 2007. Excel VBA yrityskäytössä. 1. painos. Jyväskylä: WSOYpro/Docendo-tuotteet.

Oulun ammattikorkeakoulu. N.d. Prosessiautomaatio. Pdf. Luettu 30.3.2020.
http://www.oamk.fi/~kurki/automaatiolabrat/TTT/24_Prosessiautomaatio.pdf

Shepherd, R. 2006. Excel-ohjelmointi – Tehokas hallinta. 1. painos. Jyväskylä: Gummerus Kirjapaino Oy.

Strömman, M., Hirvonen, J., Hukki, K., Tommila, T. 2007. Automaatiosuunnittelun prosessimalli. Yhteiset käsitteet verkottuneen suunnittelun perustana. Helsinki. Verkkojulkaisu 2010. https://www.automaatioseura.fi/site/assets/files/1367/automaatiosuunnittelun_prosessimalli.pdf

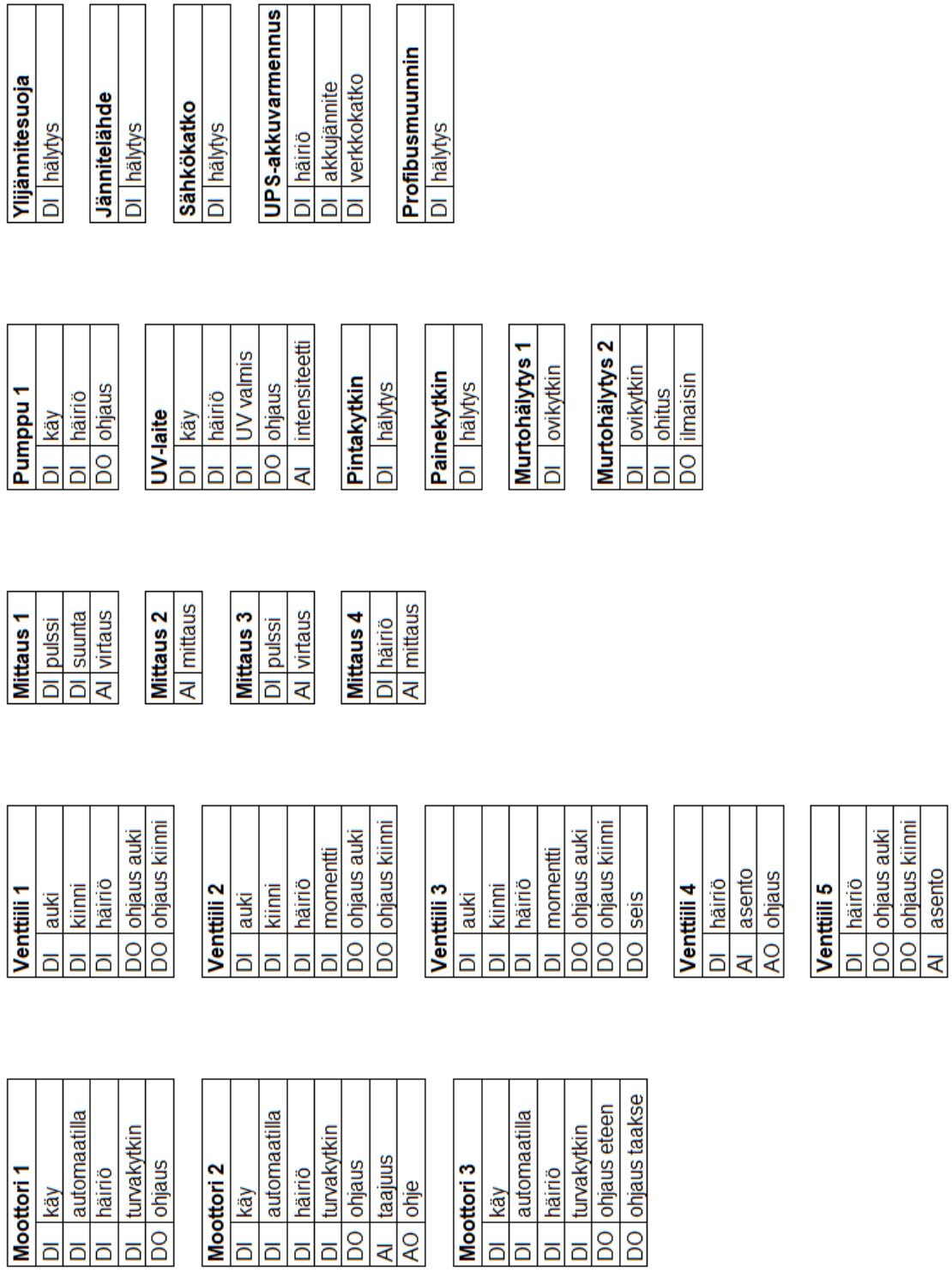
Vaihekoski, M. 2004. Rahoitusalan sovellukset ja Excel. 1. painos. Helsinki: WSOY/Oppimateriaalit.

VBA Comparison Operators. Luettu 27.1.2020. https://www.tutorialspoint.com/vba/vba_comparison_operators.htm

VBA If Statement. Luettu 27.1.2020. https://www.automateexcel.com/vba/else-if-statement

LIITTEET

Liite 1. Ohjelmoidut kenttälaitteet ja niiden I/O-määrät



									KORTTI				RISTIKYTKENTÄ						
Rivi	Positio	Tarkenne	Symboli / PLC Tag	Kommentti	I/O-tyyppi	Osoite	Tila 1 vastaa	Keskus	Tunnus	Liitin +	Liitin -	Kytkentä	Ryhmä	Liitin+	Liitin-	Piirustusnumero	Huomautus	Rev pvm	Rev kommennti
1	Z1		Z1_ALRM	Z1 Ylijännitesuoja hälytys	DI	%I0.0			A1.0		DI a.0		DIX1.0	+	1				
2	SK1		SK1_ALRM	SK1 Sähkökatko hälytys	DI	%I0.1			A1.0		DI a.1		DIX1.0	+	2				
3	G-123		G-123_ALRM	G-123 Veistorni ovikytkin hälytys	DI	%I0.2			A1.0		DI a.2		DIX1.0	+	3				
4	G-123		G-123_ACKWL	G-123 Veistorni ovikytkin ohitus	DI	%I0.3			A1.0		DI a.3		DIX1.0	+	4				
5	FIC-123		FIC-123_P	FIC-123 Tulovesi pulssi	DI	%I0.4			A1.0		DI a.4		DIX1.0	+	5				
6	FIC-123		FIC-123_DIR	FIC-123 Tulovesi suunta	DI	%I0.5			A1.0		DI a.5		DIX1.0	+	6				
7	M-123		M-123_RUN	M-123 Moottori 1 käy	DI	%I0.6			A1.0		DI a.6		DIX1.0	+	7				
8	M-123		M-123_AUTO	M-123 Moottori 1 automaatilla	DI	%I0.7			A1.0		DI a.7		DIX1.0	+	8				
9	M-123		M-123_FLT	M-123 Moottori 1 häiriö	DI	%I1.0			A1.0		DI b.0		DIX1.0	+	9				
10	M-123		M-123_SSW	M-123 Moottori 1 turvakytkin	DI	%I1.1			A1.0		DI b.1		DIX1.0	+	10				
11	V-123		V-123_OPN	V-123 Venttiili 1 auki	DI	%I2.0			A1.1		DI a.0		DIX1.1	+	1				
12	V-123		V-123_CLS	V-123 Venttiili 1 kiinni	DI	%I2.1			A1.1		DI a.1		DIX1.1	+	2				
13	V-123		V-123_FLT	V-123 Venttiili 1 häiriö	DI	%I2.2			A1.1		DI a.2		DIX1.1	+	3				
14	V-123		V-123_TRQ	V-123 Venttiili 1 momentti	DI	%I2.3			A1.1		DI a.3		DIX1.1	+	4				
15	P-123		P-123_RUN	P-123 Pumppu 1 käy	DI	%I2.4			A1.1		DI a.4		DIX1.1	+	5				
16	P-123		P-123_FLT	P-123 Pumppu 1 häiriö	DI	%I2.5			A1.1		DI a.5		DIX1.1	+	6				
17					DI	%I2.6			A1.1		DI a.6		DIX1.1	+	7				
18					DI	%I2.7			A1.1		DI a.7		DIX1.1	+	8				
19	G-123		G-123_CTRL	G-123 Veistorni ovikytkin ilmaisin	DO	%Q0.0			A1.0	DQ a.4			DOX1.0	1	-				
20	M-123		M-123_CRUN	M-123 Moottori 1 ohjaus	DO	%Q0.1			A1.0	DQ a.5			DOX1.0	2	-				
21	V-123		V-123_COPN	V-123 Venttiili 1 ohjaus auki	DO	%Q0.2			A1.0	DQ a.6			DOX1.0	3	-				
22	V-123		V-123_CCLS	V-123 Venttiili 1 ohjaus kiinni	DO	%Q0.3			A1.0	DQ a.7			DOX1.0	4	-				
23	V-123		V-123_CSTOP	V-123 Venttiili 1 seis	DO	%Q0.4			A1.0	DQ b.0			DOX1.0	5	-				
24	P-123		P-123_CTRL	P-123 Pumppu 1 ohjaus	DO	%Q0.5			A1.0	DQ b.1			DOX1.0	6	-				
25					DO	%Q1.0			A1.2	DQ a.0			DOX1.2	1	-				
26					DO	%Q1.1			A1.2	DQ a.1			DOX1.2	2	-				
27					DO	%Q1.2			A1.2	DQ a.2			DOX1.2	3	-				
28					DO	%Q1.3			A1.2	DQ a.3			DOX1.2	4	-				
29					DO	%Q1.4			A1.2	DQ a.4			DOX1.2	5	-				
30					DO	%Q1.5			A1.2	DQ a.5			DOX1.2	6	-				
31					DO	%Q1.6			A1.2	DQ a.6			DOX1.2	7	-				
32					DO	%Q1.7			A1.2	DQ a.7			DOX1.2	8	-				
33	FIC-123		FIC-123_FLW	FIC-123 Tulovesi virtaus	AI	%IW100			A1.3	0+	0-		AIX1.3	1	2				
34	M-123		M-123_FBK	M-123 Moottori 1 taajuus	AI	%IW102			A1.3	1+	1-		AIX1.3	3	4				
35					AI	%IW104			A1.3	2+	2-		AIX1.3	5	6				
36					AI	%IW106			A1.3	3+	3-		AIX1.3	7	8				
37	M-123		M-123_CRPM	M-123 Moottori 1 ohje	AO	%QW100			A1.0	0	3M		AOX1.0	1	2				
38					AO	%QW102			A1.0	1	3M		AOX1.0	3	4				
39					AO	%QW104			A1.4	0	0M		AOX1.4	1	2				
40					AO	%QW106			A1.4	1	1M		AOX1.4	3	4				