

# OOPic - Ohjelmoijan opas



Copyright(c) 1999,2000 by Savage Innovations All rights reserved.

Käännös ja muokkaus Esutech Oy 2005.

Materiaalia voi vapaasti käyttää ja levittää, kunhan Copyright-tiedot säilyvät mukana.

---

Esutech Oy  
Luotipussi 27  
40630 Jyväskylä  
FINLAND

puh: +358 400 920 139  
fax: +358 401 920 139  
email: [info@esutech.com](mailto:info@esutech.com)  
www: <http://www.esutech.com/>

Y-tunnus: 1719066-4  
ALV rek.  
kotipaikka: Jyväskylä

1	Johdanto .....	6
1.1	Mikä on OOPic? .....	6
1.2	Kuinka tämä opas on ryhmitelty.....	6
1.3	Yhteensopivuus muiden ohjelmointikielten kanssa .....	6
1.4	Termejä, jotka Sinun on hyvä tietää .....	6
1.5	Käytettävistä merkinnöistä.....	8
1.6	OOPic Software Development Kitin asennus.....	9
1.6.1	Ennen asennuksen aloittamista .....	9
1.6.2	Laitteistovaatimukset .....	9
1.6.3	Lue README-tiedosto .....	9
1.6.4	Asennus .....	9
1.7	Esimerkkisovellukset .....	10
1.8	On Line ohje.....	10
2	Ensimmäinen OOPic sovelluksesi.....	11
2.1	Mikä on OOPic-sovellus? .....	11
2.2	OOPic-sovelluksen teon eri vaiheet.....	11
2.3	OOPic Software Development Kitin käynnistys .....	11
2.4	Ensimmäisen OOPic-sovelluksen rakenne .....	11
2.5	Laitteistoliitynnän suunnittelu.....	12
2.6	Komponenttien kytkentä .....	13
2.7	Ohjelman lataaminen ja ajaminen.....	14
3	Ensimmäinen Virtuaalipiiri.....	15
3.1	Mikä on Virtuaalipiiri? .....	15
3.2	Ensimmäisen Virtuaalipiirisovelluksen teon vaiheet.....	15
3.3	OOPic sovelluksen tekeminen.....	15
3.4	Tunnista toiminnot, jotka voidaan toteuttaa virtuaalipiireillä .....	15
3.5	Tee virtuaalipiirin kytkentäkaavio .....	16
3.6	Lisää ja linkitä virtuaalipiiri sovellukseen .....	16
3.7	Virtuaalipiirisovelluksen lataaminen ja ajaminen .....	18
4	Ensimmäinen tapahtumaohjattu ohjelma .....	19
4.1	Mikä on tapahtumaohjattu ohjelma .....	19
4.2	Ensimmäisen tapahtumaohjetun sovelluksen teon vaiheet.....	19

4.3	OOPic sovelluksen tekeminen.....	19
4.4	Tunnista toiminnot, jotka voidaan toteuttaa tapahtumina .....	19
4.5	Lisää Event (tapahtuma)-olio sovellukseen .....	19
4.6	Tee Event (tapahtuma)-olion aliohjelma.....	21
4.7	Tapahtumaohjatun sovelluksen lataaminen ja ajaminen .....	22
5	Ohjelmoinnin perusteet .....	23
5.1	OOP .....	23
5.2	OOPic sovelluksen rakenne.....	23
5.3	Kuinka tapahtumaohjattu sovellus toimii .....	24
5.3.1	Tapahtumaohjattu vs. Perinteinen ohjelmointi .....	25
5.3.2	Ohjelmakoodi, joka suoritetaan OOPicin käynnistyessä .....	25
5.3.3	Tapahtumaohjatun ohjelman lopetus .....	26
5.4	Oliot .....	26
5.5	Muuttujat .....	27
5.6	Aliohjelmat .....	27
5.7	Funktiot .....	28
5.8	Ohjausrakenteet .....	28
5.8.1	Vuonohjaus .....	28
5.8.2	Valintarakenteet .....	28
5.8.3	Silmukkarakenteet .....	29
5.9	Kommenttien lisääminen ohjelmakoodiin.....	29
6	Johdanto olio-ohjelmointiin .....	30
6.1	Mitä on OO?.....	30
6.2	Miksi olioita?.....	30
6.3	Mitä olioilla voi tehdä?.....	30
6.4	Mistä oliot tulevat? .....	31
6.5	Olioiden luominen .....	32
6.5.1	Olioiden nimet .....	32
6.5.2	Taulukoiden luominen olioista .....	33
6.5.3	Olioiden luominen luokista, joiden koko on muuttuva .....	33
6.6	Olioiden kanssa työskentely.....	34
6.7	Olion ominaisuuksien käsittely .....	34
6.7.1	Ominaisuuden arvon asettaminen .....	34

6.7.2	Ominaisuuksien arvojen lukeminen.....	35
6.7.3	Oletusominaisuuksien käyttö .....	36
6.8	Toimintojen suorittaminen metodien avulla .....	36
6.8.1	Metodien käyttö ohjelmakoodissa .....	36
6.9	Tapahtumien käsittely .....	37
6.9.1	oEvent-olio .....	37
6.10	Olioiden yhteen linkittäminen .....	37
6.11	Dynaaminen tiedonsiirto (Dynamic Data Exchange) .....	38
7	Muuttujat ja vakiot .....	39
7.1	Mitä ovat muuttujat?.....	39
7.2	oBit, oNibble, oByte ja oWord muuttujat.....	39
7.3	oBuffer-muuttuja .....	40
7.4	oEEProm-muuttuja .....	40
7.5	oRam-muuttuja .....	40
7.6	Vakiot .....	40
8	Liittäminen laitteisiin .....	41
8.1	OOPicin laitteistoliityntä .....	41
8.2	Laitteisto-olioiden käyttö .....	41
9	Virtuaalipiirit .....	42
9.1	Mikä on virtuaalipiiri? .....	42
9.2	Kuinka virtuaalipiirit tehdään?.....	42
9.3	Olioiden linkittäminen .....	42
9.4	Osoittimen ominaisuudet.....	42
9.5	Olio-osoitimet.....	43
9.6	Lippuosoittimet .....	43
10	OOPic järjestelmäolio .....	45
10.1	Mikä on OOPic-olio? .....	45
10.2	OOPicin käyttöjärjestelmän hallinta.....	45
10.3	OOPicin käyttöjärjestelmän tilan lukeminen.....	45
10.4	Muiden OOPicin toimintojen hallinta .....	45
11	OOPicin matematiikka .....	47
11.1	OOPicin matematiikasta .....	47
11.2	Lausekkeet.....	47

11.3	Operaattorit .....	47
11.4	Operaatioiden suoritusjärjestys .....	47
11.5	Aritmeettiset operaattorit .....	48
11.6	Loogiset Operaattorit .....	48
11.7	Vertailuoperaattorit.....	49
11.8	Funktiot .....	49
12	Dynaaminen tiedonsiirto ( <i>DDE</i> ).....	50
12.1	Mitä on Dynaaminen tiedonsiirto? .....	50
12.2	oDDELink-olio.....	50
12.3	DDE-virtuaalipiiri .....	50
12.4	Verkon solmut.....	51
12.5	DDE-keskustelu.....	51
12.6	DDE-isäntä .....	52
12.6.1	Kuinka isännän oDDELink-olio toimii .....	52
12.7	DDE-orja .....	53
12.7.1	Kuinka orjan oDDELink toimii.....	54
12.8	I2C-verkon kytkennät .....	54
13	Merkkijonot ( <i>Strings</i> ) .....	56
13.1	Mitä ovat merkkijonot? .....	56
13.2	Merkkijonojen käyttö .....	56
13.3	Merkkijonojen käyttö oSerial- ja oSerialPort-olioiden kanssa.....	56
13.4	Merkkijonojen käyttö oDataStrobe -olion kanssa .....	57
13.5	Merkkijonojen käyttö oBuffer-olion kanssa .....	57

# 1 Johdanto

## 1.1 Mikä on OOPic?

OOPic on akronyymi sanoista Object-Oriented Programmable Integrated Circuit eli olio-ohjelmitava integroitu piiri.

OOPicin olio-pohjainen ohjelmointikieli on suunniteltu helpottamaan elektronisten komponenttien liittämistä OOPiciin. Laitteistoliityntä suunnitellaan ohjelmistossa siten, että luodaan tarvittavat oliot ja asetetaan niiden ominaisuudet halutun toiminnon aikaansaamiseksi laitteistossa. Oliot voivat liittyä myös virtuaalipiiriin. Tällöin laitteistoliityntä ja siihen kytketty virtuaalipiiri reagoivat laitteistossa tapahtuviin tapahtumiin ohjelmassa määritetyllä tavalla.

Tässä kappaleessa kerrotaan kuinka OOPic Software Development Kit asennetaan tietokoneeseen ja esitellään oppaan muut osat.

## 1.2 Kuinka tämä opas on ryhmitelty

Tämän oppaan kappaleet voidaan ryhmitellä kolmeen osaan:

### Osa 1

Kappaleet 1-4 kuvaavat ensiaskeleet OOPicin käytöstä kuten ohjelman asennuksen ja kolme yksinkertaista ohjelmaa, joiden avulla pääset tutustumaan OOPic Basiciin ja sen käyttöön OOPicin kanssa.

### Osa 2

Kappaleissa 5-8 käydään läpi OOPicin ohjelmoinnin perusteita kuten oliot, muuttujat, ohjelman hallinta ja laitteistotason I/O.

### Osa 3

Kappaleissa 9-12 käydään läpi kehittyneempiä asioita kuten virtuaalipiirit, OOPic järjestelmäolio, matemaattiset operaatiot ja verkottaminen.

## 1.3 Yhteensopivuus muiden ohjelmointikielten kanssa

Alla kuvataan OOPicin ohjelmointikielten yhteensopivuutta verrattuna muihin yleisiin ohjelmointikieliin.

BASIC: OOPic Basic syntaksi on 100% yhteensopiva Microsoftin Visual Basic kielen kanssa. Kielen perussyntaksi pohjautuu suoraan perinteisestä Basic syntaksista ja on siten tuttua aivan aloittelevillekin Basic-ohjelmoijille.

C: OOPic C on vielä osittain kehitysvaiheessa.

Java: OOPic Java on vielä osittain kehitysvaiheessa.

## 1.4 Termejä, jotka Sinun on hyvä tietää

Seuraavassa luettelossa on määritelty joitain yleisesti OOPicin manuaaleissa määritettyjä termejä.

Avainsana ( <i>Keyword</i> )	Mikä tahansa varattu sana OOPicin kielessä. Avainsanat voivat olla komentoja, funktioita, lauseita tai mitä tahansa sanoja, joita OOPic käyttää jossakin muussa käytössä. OOPic ei erottele isoja ja pieniä kirjaimia toisistaan.
Tunniste ( <i>Identifier</i> )	Sana, jonka ohjelmoija määrittelee kuvaamaan sovellukseen liittyvää asiaa. Tunniste ei saa olla sama kuin varattu sana ja sen tulee alkaa kirjaimella.
Vakio ( <i>Constant</i> )	Ennalta määritetty avainsana, joka edustaa jotain arvoa. Vakioita käytetään numeroiden sijaan, että ohjelmakoodi olisi selkeämmin luettavissa.
Muuttuja ( <i>Variable</i> )	Tunniste, jolla on arvo, mutta toisin kuin vakiolla, sen arvo voidaan määrittää. Muuttujan arvo säilyy samana kunnes sille annetaan uusi arvo.
Komento ( <i>Command</i> )	Avainsana, joka käskee OOPicin suorittamaan tietyn toiminnon.
Funktio ( <i>Function</i> )	Avainsana, joka käskee OOPicin suorittamaan tietyn laskuoperaation ja palauttaa vastauksen. Funktio muistuttaa vakiota siinä, että sillä on arvo, mutta arvo lasketaan uudelleen aina kun funktio suoritetaan.
Lauseke ( <i>Statement</i> )	Lauseke muodostuu komennoista, funktioista, tunnisteista ja vakioista, jotka saavat OOPicin suorittamaan tietyn toiminnon ja kuvaavat kuinka se tehdään. Lauseke yleensä alkaa komennolla, mutta se voi alkaa myös tunnisteella.
Operaattori ( <i>Operator</i> )	Operaattorit ovat komentoja, jotka suorittavat laskentaa, vertailuja ja sijoituksia.

Jos et ole aiemmin harrastanut olio- tai tapahtumapohjaista ohjelmointia, voi osa terminologiasta tuntua alussa hieman oudolta. Alla on määritelty joitain yleisiä termejä, joita OOPicin manuaaleissa käytetään.

Tapahtuma ( <i>Event</i> )	Toiminto, jonka olio tunnistaa ja jolle voidaan kirjoittaa koodi, joka suoritetaan tapahtuman sattuessa. Tapahtuma voi aiheutua laitteistossa tapahtuvan ilmiön seurauksena (esim. jännitetason muutos) tai ohjelmakoodin aiheuttamana.
Tapahtumaohjattu ( <i>Event-driven</i> )	Termi, jota käytetään kuvaamaan erästä ohjelmointimallia. Toisin kuin perinteisessä peräkkäin suoritettaviin käskyihin perustuvassa ohjelmoinnissa tapahtumaohjattu sovellus on valmiustilassa kunnes jokin olio reagoi tapahtumaan ja suorittaa siihen liittyvän koodin.
Metodi ( <i>Method</i> )	Avainsana (samanlainen kuin funktio tai lauseke),

	joka on osa oliota ja jonka toiminto vaikuttaa kyseiseen olioon.
Olio ( <i>Object</i> )	Termi, jolla kuvataan kokonaisuutta, joka koostuu muuttujista ja ohjelmakoodista ja joka toimii yhtenä loogisena yksikkönä.
Oliopohjainen ( <i>Object-oriented</i> )	Termi, jota käytetään kuvaamaan ohjelmointimallia, jossa muuttujista ja koodista kootaan loogisia kokonaisuuksia, olioita.
Ominaisuus ( <i>Property</i> )	Muuttuja, joka on osa oliota.

Virtuaalipiirit (*Virtual Circuits*) on tapa emuloida elektronisten piirien toimintaa mikrokontrollerilla. Alla on määritelty joitain yleisiä termejä, joita OOPicin manuaaleissa käytetään puhuttaessa virtuaalipiireistä.

Oletusarvo ( <i>Default Value</i> )	Termi, jota käytetään kuvaamaan olion ominaisuutta, jota käytetään, jos ohjelmassa ei ole mitään muuta arvoa ominaisuudelle määritetty.
Lippu ( <i>Flag</i> )	Termi, jota käytetään kuvaamaan 1-bittistä ominaisuutta, joka voidaan linkittää lippuosoittimeen ( <i>Flag-Pointer</i> ).
Linkki ( <i>Link</i> )	Termi, jota käytetään kuvaamaan virtuaalipiirin liityntäpistettä.
Osoitin ( <i>Pointer</i> )	Termi, jota käytetään kuvaamaan ominaisuutta, joka osoittaa toiseen ominaisuuteen.
Virtuaalipiiri ( <i>Virtual Circuit</i> )	Virtuaalipiiri on ohjelmallinen piiri OOPicin sisällä, joka näyttää ulospäin elektroniikkapiiriltä, mutta on todellisuudessa OOPicin käyttöjärjestelmän päällä pyörivä ohjelma, joka emuloi piirin toimintaa.

### 1.5 Käytettävistä merkinnöistä

Asiat, jotka on hakasulkeissa, [ ], ovat valinnaisia eli joita voidaan käyttää mutta ei ole välttämätöntä. Jos hakasulkeissa olevan asian perässä on kolme pistettä [exp...], se tarkoittaa, että parametrejä voidaan antaa ihan kuinka monta tahansa, mutta voidaan olla antamatta yhtään.

Seuraavassa on lueteltuna tässä oppaassa käytetyt lyhenteet:

{muuttuja}	Mikä tahansa muuttuja.
{olio}	Mikä tahansa olio.
{luokka}	Olion luokka.

{otsake}	Mikä tahansa otsake.
{vakio luettelo}	Lista tietoalkioista.
{subscript}	Numeerinen arvo.
{operaattori}	Aritmeettinen tai looginen operaattori.
{lauseke}	Aritmeettinen tai looginen lauseke.

## 1.6 OOPic Software Development Kitin asennus

### 1.6.1 Ennen asennuksen aloittamista

Ennenkuin aloitat OOPic Software Development Kitin asennuksen tarkista, että tietokoneesi täyttää minimilaitteistovaatimukset ja että tarkistat README-tiedosta uusimmat muutokset ja korjaukset.

### 1.6.2 Laitteistovaatimukset

Käyttääksesi OOPic Software Development Kitiä tietokoneessasi täytyy olla vähintään:

- IBM®-yhtensopiva tietokone, jossa Windows® 95® tai myöhempi käyttöjärjestelmä.
- Kiintolevyllä vähintään 5 megatavua tyhjää tilaa.
- Hiiri tai muu osoitinlaite.
- Vapaa kirjoitinportti, jos käytät kirjoitinporttiin kytkettävää ohjelmointikaapelia.

### 1.6.3 Lue README-tiedosto

README-tiedosto sisältää muutokset OOPicin dokumentaatiossa niiden julkaisemisen jälkeen. Tarkista tiedoston ensimmäisestä osiosta yksityiskohdat ja uudet tiedot.

### 1.6.4 Asennus

OOPic Software Development Kit asennetaan tietokoneeseen käyttäen Setup-ohjelmaa. Setup-ohjelma asentaa OOPic Software Development Kitin, ohjetiedostot ja esimerkkisovellukset.

Tärkeää: Et voi suorittaa asennusta pelkästään kopioimalla tiedostot asennuslevykkeiltä kiintolevyille. Sinun täytyy käyttää asennusohjelmaa, joka purkaa pakatut tiedostot oikeisiin hakemistoihin.

Kun suoritat asennuksen Setup ohjelmalla, OOPicille luodaan hakemisto. Kaikki tarvittavat tiedostot asennetaan tämän hakemiston alle lukuunottamatta Microsoft® Visual Basic® runtime -tiedostoja.

Asenna OOPic Software Development Kit tietokoneeseesi seuraavasti:

Paina Windowsin Start-painiketta.

Valitse Run Windowsin Start Menusta.

Kirjoita "A:setup" ja paina return.

Seuraa asennusohjelman antamia ohjeita.

### **1.7 Esimerkkisovellukset**

Dokumentaation lisäksi OOPic Software Development Kit sisältää esimerkkisovelluksia, jotka voit ladata OOPiciin. Nämä sovellukset ovat erinomaista opiskelumateriaalia. Voit kopioida minkä tahansa osan niistä omiin sovelluksiisi ja muokata tarpeen mukaan. Lisätietoja näistä löytyy Internetistä osoitteesta: <http://www.oopic.com>

### **1.8 On Line ohje**

Ohjeet löytyvät valitsemalla Contents-komento Help-valikosta.

Nopein tapa löytää jokin tietty aihe on käyttää Search(Etsi) keskusteluikkunaa.

- Help-valikosta valitse "Search for Help On", tai paina "Search" painiketta missä tahansa ohjeikkunassa.
- In the Search dialog box, type a word, or select one from the list by scrolling up or down. Press ENTER or choose Show Topics to display a list of topics related to the word you specified.
- Select a topic name, and the press ENTER or choose "GO TO" to view the topic.

OOPic Internet Support Services

- Several services are available on our Internet world-wide-web page.
- You can access the OOPic Internet Support Services at:  
<http://www.oopic.com>

## 2 Ensimmäinen OOPic sovelluksesi

### 2.1 Mikä on OOPic-sovellus?

OOPic sovellus on tietokoneohjelma, joka on kirjoitettu ohjaamaan OOPic mikrokontrolleriin kytkettyjä elektronisia laitteita. Itse ohjelma kirjoitetaan, tallennetaan ja käännetään PC-tietokoneessa OOPicin käskyiksi ja ladataan OOPic-mikrokontrolleriin. Kun ohjelma on ladattu OOPiciin, se suoritetaan siinä ja PC:tä ei enää suoritusvaiheessa tarvita.

Tässä kappaleessa käydään läpi pienen sovelluksen toteutus. Tarkoitus on tutustuttaa uudet käyttäjät OOPic-kehitysympäristöön ja ohjelman kehityksen eri vaiheisiin.

### 2.2 OOPic-sovelluksen teon eri vaiheet

Sovelluksen teossa OOPicille on 5 eri vaihetta

- OOPic Software development kitin käynnistys
- Laitteistoliitynnän suunnittelu
- Ohjelmakoodin kirjoitus
- Laitteiston kytkentä
- Sovelluksen lataus ja suoritus

### 2.3 OOPic Software Development Kitin käynnistys

Valitsemalla "OOPic Language Compiler" Windowsin Käynnistä-valikosta käynnistää OOPic Software Development Kitin. Kun OOPic Software Development Kit avautuu, se käynnistyy koko näytön kokoisena sovelluksena, jossa on joukko valikoita, ohjelmaeditorialue, jossa kirjoitat ohjelman ja luettelo käytetyistä olioista.

Valikot sisältävät operaatiot, joilla mm. talletat ja lataat kirjoittamasi ohjelmasi levyille ja käännet ja siirrä ohjelmasi OOPic mikro-ohjaimeen.

### 2.4 Ensimmäisen OOPic-sovelluksen rakenne

Ensimmäinen OOPic-sovellus on vilkkuva led (*light emitting diode*). Kytkentä sytyttää ja sammuttaa ledin kerran sekunnissa.

Tarvittavat osat ovat:

- 1 kpl OOPic mikrokontrolleri
- 1 kpl Led
- 1 kpl 220Ω vastus
- 1 kpl OOPiciin kytkettävä patteri
- 1 OOPic ohjelmointikaapeli

## 2.5 Laitteistoliitynnän suunnittelu

OOPic sovelluksessa laitteistoliityntä on joukko oliota, jotka on määritelty toimimaan OOPiciin fyysisesti kytkettyjen laitteiden kanssa.

Ensimmäisessä sovelluksessa LED vilkkuu koko ajan. Tämä toteutetaan kytkemällä ja katkaisemalla sen I/O-linjan jännitettä, johon LED on kytketty. Tätä varten tarvitaan yksi digitaalinen I/O-linja. Kirjoita ensimmäiseksi koodiriviksi ohjelmaan seuraava lauseke:

```
Dim LED As New oDio1
```

Tällöin OOPic luo ilmentymän oDio1 (1-bittinen digitaalinen I/O-linja)-oliosta nimelle "LED". oDio1-olio voidaan luokitella "Laitteisto-olioksi" (*Hardware Object*). Tämä tarkoittaa, että olio toimii OOPicin I/O-linjan kanssa ennaltamääritellyllä tavalla. oDio1-olio joko asettaa jonkin OOPicin I/O-linjan jännitteen halutuksi (0 tai 5 V) tai lukee I/O-linjassa olevan jännitteen.

Kun olion ilmentymä on luotu, kaikki viittaukset kyseiseen ilmentymään tapahtuvat kirjoittamalla sen nimi, piste ja sen ominaisuuden tai metodin nimi, johon halutaan viitata.

Määrittääksemme mitä I/O-linjaa LED-olio käyttää jännitelähtönään, tulee asettaa joitain sen ominaisuuksia. Lisää seuraavat lausekkeet ohjelman loppuun:

```
Sub Main()  
    LED.IOLine = 31  
    LED.Direction = cvOutput  
End Sub
```

Lausekkeet SUB MAIN() ja END SUB määrittävät alun ja lopun proseduurista nimeltä MAIN. Proseduurin nimeltä "MAIN" on ensimmäisenä suoritettava proseduurin, kun OOPic käynnistetään tai resetoitetaan.

Rivi "LED.IOLine = 31" asettaa IOLine ominaisuuden oliolle nimeltä LED arvoon 31 ja rivi "LED.Direction = cvOutput" asettaa Direction (*Suunta*) ominaisuuden arvoksi 0 (cvOutputin arvo). Näiden kahden lausekkeen suorituksen jälkeen arvo, joka kirjoitetaan LED.Value arvoksi tulee I/O-linjaan 31. Jos Value ominaisuus LED oliolla asetetaan 0:ksi, 0 voltia tulee I/O-linjan 31 jännitteeksi ja jos se asetetaan 1:ksi, 5 voltia tulee I/O-linjan 31 jännitteeksi.

Vaihtaaksemme LED.Value ominaisuutta 1 vilkahduksen sekuntitahdilla lisäämme koodin, joka koko ajan asettaa sen arvoksi OOPic.Hz1 arvon. (OOPic.Hz1 on arvo, joka vaihtuu kerran sekunnissa). Ohjelman loppuun, juuri ennen "END SUB":a lisätään seuraavat rivit:

```
Do  
    LED.Value = OOPic.Hz1
```

## Loop

Ensimmäinen lauseke asettaa silmukan alun. Toinen lauseke asettaa LED-olion Value-ominaisuudeksi OOPic.Hz1:n arvon, joka vaihtuu kerran sekunnissa. Kolmas lauseke päättää silmukkarakenteen ja aiheuttaa ohjelman siirtymisen Do lausekkeeseen.

LED-olion Value-ominaisuuden asettaminen OOPic.Hz1:ksi silloin kun se on 1 aiheuttaa IOLine:n asettumisen 5 volttiin ja vastaavasti asettaminen OOPic.Hz1:ksi silloin kun se on 0 aiheuttaa IOLine:n asettumisen 0 volttiin.

Käytettäessä Do...Loop rakennetta sijoituslausekkeen ympärillä sijoituslauseketta toistetaan loputtomiin. Tuloksena tästä on, että LED syttyy ja sammuu kerran sekunnissa.

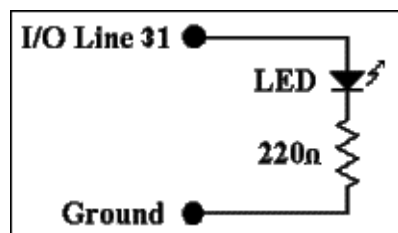
Seuraava koodi näyttää kokonaisen edellisten kappaleiden mukaisen listauksen.

```
Dim LED As New oDio1
```

```
Sub Main()  
    LED.IOLine = 31  
    LED.Direction = cvOutput  
    Do  
        LED.Value = OOPic.Hz1  
    Loop  
End Sub
```

## 2.6 Komponenttien kytkentä

Kytke led yhteen vastuksen kanssa I/O-linjaan 31 ja maahan(*ground*) kuten kuvassa.



Sekä I/O-linja 31 että maa löytyvät OOPicin 40-pinnisestä I/O-liittimestä. OOPicin 40-pinnisen I/O-liittimen pinnijärjestys löytyy mm.: <http://www.oopic.com/con40.htm>

## 2.7 Ohjelman lataaminen ja ajaminen

F5-näppäimen painaminen aloittaa ohjelman kääntämisen ja "oex"-tiedoston lataamisen OOPiciin. Jos käännettäessä havaitaan virheitä, ne ilmoitetaan ja kääntäminen keskeytyy. Muutoin kääntäjä tekee ".oex"-päätteisen (OOPic Executable) tiedoston. Jos et ole tallentanut ohjelmaa tai olet tehnyt siihen muutoksia viimeisen tallennuksen jälkeen, kääntäjä pyytää sinua tallentamaan työsi. Tässä tapauksessa nimeä työsi nimelle EkaHarj. Kirjoita "EkaHarj" "File name"-kohtaan "Save As"-ikkunassa ja paina Enter.

Jotta suoritettava ohjelma voidaan siirtää OOPiciin, ohjelmointikaapelin tulee olla kytketty PC:n ja konfiguroitu oikein. Sen toisen pään tulee olla kytkettynä OOPicin ohjelmointiliittimeen ja virtalähde tulee olla kytkettynä OOPiciin. Katso kohdasta "Ohjelmointikaapeli" lisätietoja.

## 3 Ensimmäinen Virtuaalipiiri

### 3.1 Mikä on Virtuaalipiiri?

Virtuaalipiiri (*Virtual Circuit*) on OOPic:n toiminto, joka näyttää toiminnallisesti diskreetiltä elektroniikkapiiriltä, mutta on itse asiassa OOPic käyttäjärjestelmä, joka emuloi piirin toimintaa.

Virtuaalipiirit tehdään ohjelmallisesti yhdistämällä toisiinsa OOPic oliot samalla tavalla kuin tehtäisiin yhdistettäessä toisiinsa fyysisesti elektroniikkakomponentteja. Jokaista yksittäistä muodostetun virtuaalipiirin osaa voidaan muokata tai sen arvoa käyttää missä tahansa ohjelman osassa.

Virtuaalipiirejä käytetään jatkuvien prosessien ohjaamiseen. Esimerkkinä tällaisesta on sähköpiiri, jossa on valokatkaisin ja polttimo. Valokatkaisin ohjaa jatkuvasti polttimon palamista. OOPic sovelluksessa asiat, joita jatkuvasti päivitetään tai valvotaan, voidaan toteuttaa virtuaalipiireillä.

### 3.2 Ensimmäisen Virtuaalipiirisovelluksen teon vaiheet

Virtuaalipiirisovelluksen luomisessa on 4 vaihetta

1. Tee OOPic sovellus
2. Tunnista toiminnot, jotka voidaan toteuttaa virtuaalipiireinä
3. Tee virtuaalipiirin kytkentäkaavio
4. Lisää ja linkitä virtuaalipiiri sovellukseen

### 3.3 OOPic sovelluksen tekeminen

Kappaleessa 2 tehtiin OOPic sovellus, joka vilkuttaa LEDiä jatkuvasti, kun OOPic.Hz1-ominaisuus kytkettiin oDio1-olion arvoksi Value-ominaisuuteen. OOPic.Hz1 arvo vaihtuu 0:sta 1:n ja takaisin kerran sekuntissa. Kytkettäessä oDio1-olion I/O-linjaan LED tämä vilkkuu jänniteen vaihtelun tahdissa. Tuloksena oli sovellus, joka vilkutti LEDiä.

Tässä kappaleessa käydään läpi kuinka kappaleen 2 sovellusta muokataan siten, että virtuaalipiiriä käytetään hyväksi sovelluksessa.

### 3.4 Tunnista toiminnot, jotka voidaan toteuttaa virtuaalipiireillä

Kappaleen 2 sovelluksessa prosessi, joka koko ajan siirtää OOPic.Hz1:n arvon oDio-olion arvokse, vie ohjelman suoritusaikaa. Tämän prosessin toiminta on samanlaista kuin valokatkaisijasta ja polttimosta muodostuvan sähköpiirin toiminta.

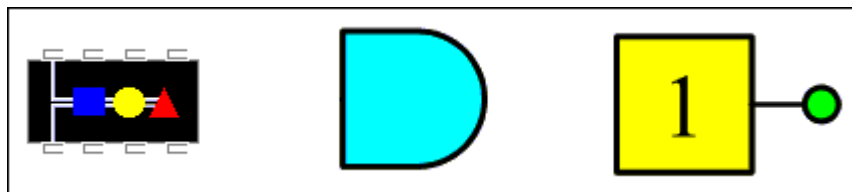
Yksi OOPic:n käsittelyolioista, oGate olio, voi tehdä saman toiminnon kuin OOPic sovellus tekee nyt ohjelmakoodilla, joten muokattaessa ohjelmaa käyttämään virtuaalipiiriä tämän olion kanssa, jää varsinaisessa ohjelmassa aikaa tehdä muita toimintoja.

### 3.5 Tee virtuaalipiirin kytkentäkaavio

Luotaessa virtuaalipiiriä paras tapa kuvata sen toiminnot on piirtää kytkentäkaavio. Toisin kuin ohjelmassa, missä käytetään vuokaaviota kuvaamaan ohjelman toimintaa, virtuaalipiirin toimintaa kuvataan parhaiten kytkentäkaaviolla.

Ensimmäisessä virtuaalipiirisovelluksessaamme oGate-oliota käytetään siirtämään OOPic.Hz1:n arvo oDio1-olion arvoksi.

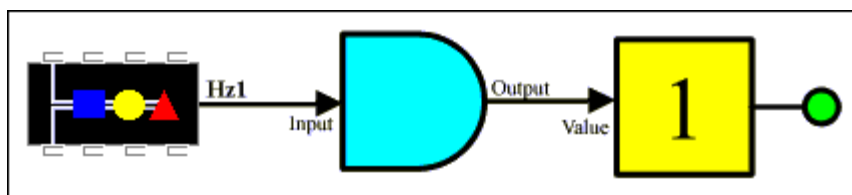
KytKentäkaavion teko aloitetaan piirtämällä OOPici-olion, oGate-olion ja oDio1-olion kuvakkeet.



Seuraavaksi piirretään nuolet olioiden niiden ominaisuuksien välille, jotka kuvaavat tiedon siirtoa olioiden välillä.

Ensin piirretään nuoli OOPic-oliosta oGate-olioon. Tämä nimetään "Hz1"-nimellä OOPic-olion päässä ja "Input"-nimellä oGate-olion päässä.

Seuraavaksi piirretään nuoli oGate-oliosta oDio1-olioon. Tämä nimetään "Output"-nimellä oGate-olion päässä ja "Value"-nimellä oDio1-olion päässä.



### 3.6 Lisää ja linkitä virtuaalipiiri sovellukseen

Virtuaalipiiri lisätään OOPic sovellukseen ohjelmakoodiriveillä.

Koska tämä sovellus perustuu kappaleessa 2 tehtyyn sovellukseen, otetaan sen koodi pohjaksi sovellusta tehtäessä. Kappaleen 2 sovelluksen lopullinen muoto oli seuraava:

```
Dim LED As New oDio1
```

```
Sub Main()
```

```
    LED.IOLine = 31
```

```
    LED.Direction = cvOutput
```

```
    Do
```

```
        LED.Value = OOPic.Hz1
```

```
Loop
```

```
End Sub
```

Kuten aiemmin mainittiin, virtuaaliopiiri käyttää oGate-oliota tekemään samaa toimintoa, joka OOPic sovelluksessa tehdään ohjelmakoodilla. Lisätään ohjelman alkuun seuraava rivi:

```
Dim WIRE As New oGate
```

Tällä luodaan uusi ilmentymä oGate-oliosta nimelle "WIRE". oGate-olio on luokiteltu "käsittelyolioksi" (*Processing Object*). Tämä tarkoittaa sitä, että olio käsittelee toisten olioiden arvoja jollakin ennalta määritellyllä tavalla. oGate-olion toiminta on lukea arvoja yhdestä tai useammasta oliosta, suorittaa niillä määritetty looginen operaatio ja tallettaa operaation tulos jonkin olion ominaisuuden arvoksi.

Sovelluksen Do...Loop rakenne voidaan poistaa, koska se tullaan korvaamaan virtuaaliopiirillä. Alkaen Do lauseesta poista seuraavat 3 koodiriviä.

```
Do
    LED.Value = OOPic.Hz1
Loop
```

Korvaa ne seuraavalla kolmella koodirivillä.

```
WIRE.Input1.Link(OOPic.Hz1)
WIRE.Output.Link(LED.Value)
WIRE.Operate = cvTrue
```

Ensimmäinen rivi, "WIRE.Input1.Link(OOPic.Hz1)", linkittää "WIRE"-olion Input1-ominaisuuteen OOPic.Hz1-ominaisuuden. Toinen rivi, "WIRE.Output.Link(LED.Value)", linkittää "WIRE"-olion Output-ominaisuuden "LED"-olion Value ominaisuuteen. Ja kolmas rivi, "WIRE.Operate = cvTrue", asettaa "WIRE"-olion Operate-ominaisuuden arvoksi 1 (cvTrue arvoksi), mikä käynnistää olion toiminnan.

Kun ohjelma suorittaa nämä 3 koodiriviä, virtuaaliopiiri toimii seuraavasti: OOPic.Hz1-ominaisuuden arvo luetaan oGate-olioon.

oGate suorittaa loogisen OR (TAI) operaation muiden tulojen kanssa. Tässä sovelluksessa ei ole käytössä kuin yksi tulo joten lähdön tulos on aina sama kuin tulon arvo.

LED.Value arvoksi asetetaan loogisen OR-operaation tulos.

Tämä proseduuri toimii taustalla ja jatkaa toimintaansa niin pitkään kunnes WIRE.Operate-ominaisuuden arvo asetetaan takaisin 0:ksi (cvOff).

Seuraavassa ohjelmakoodissa on kaikki edellisissä kappaleissa tehdyt muutokset mukana.

```
Dim WIRE As New oGate
```

```
Dim LED As New oDio1
```

```
Sub Main()
```

```
    LED.IOLine = 31
```

```
    LED.Direction = cvOutput
```

```
    WIRE.Input1.Link(OOPic.Hz1)
```

```
    WIRE.Output.Link(LED.Value)
```

```
    WIRE.Operate = cvTrue
```

```
End Sub
```

### **3.7 Virtuaaliipiirisovelluksen lataaminen ja ajaminen**

Virtuaaliipiirisovelluksen lataaminen ja ajaminen ei eroa muiden OOPic sovellusten ajamisesta.

F5-näppäimen painaminen kääntää ohjelman ja lataa "oex" tiedoston OOPiciin. Jos virheitä huomataan, kääntäminen keskeytyy ja virheet ilmoitetaan näytössä. Muutoin luodaan "oex" (OOPic Executable) tiedosto. Jos et ole tallettanut ohjelmaa aikaisemmin tai olet muuttanut sitä viimeisen talletuksen jälkeen, kääntäjä pyytää sinua tallettamaan ohjelman. Jos kyseessä on ensimmäinen talletus, kääntäjä kysyy ohjelmatiedoston nimeä. Nimeä tämä ohjelma nimelle "EkaVC".

Jotta suoritettava ohjelma voidaan siirtää OOPiciin, ohjelmointikaapelin tulee olla kytketty PC:n ja konfiguroitu oikein. Sen toisen pään tulee olla kytkettynä OOPicin ohjelmointiliittimeen ja virtalähde tulee olla kytkettynä OOPiciin. Katso kohdasta "Ohjelmointikaapeli" lisätietoja.

## 4 Ensimmäinen tapahtumaohjattu ohjelma

### 4.1 Mikä on tapahtumaohjattu ohjelma

Tapahtuma on määritelty miksi tahansa tilan muutokseksi, jonka olio tunnistaa. Tapahtumaohjattu ohjelma (*Event Driven Program*) on ohjelma, jossa mikä tahansa tilan muutos voi aiheuttaa aliohjelman suorituksen vaikka sitä ei olisi pääohjelmasta kutsuttukaan. Tällöin koodin suorituserjestys riippuu siitä, mitä tapahtumia järjestelmässä tapahtuu.

Tapahtumaohjatuissa ohjelmissa OOPicin oEvent-olio kutsuu tapahtuman käsittelevää koodia.

### 4.2 Ensimmäisen tapahtumaohjetun sovelluksen teon vaiheet

Tapahtumaohjatun OOPic sovelluksen luomisessa on 4 vaihetta

1. Tee OOPic sovellus
2. Tunnista toiminnot, jotka voidaan toteuttaa tapahtumina
3. Lisää Event (tapahtuma)-olio sovellukseen
4. Tee Event-olion aliohjelma

### 4.3 OOPic sovelluksen tekeminen

Kappaleessa 3 tehtiin sovellus, joka vilkuttaa LEDiä käyttäen oGate-oliota liittämällä OOPic.Hz1-ominaisuus oDio1-olion Value-ominaisuuteen. Tässä kappaleessa muokataan kappaleen 3 sovelluksesta tapahtumaohjattu.

### 4.4 Tunnista toiminnot, jotka voidaan toteuttaa tapahtumina

Kappaleessa 3 virtuaalipiiri sijoitti jatkuvasti OOPic.Hz1 arvon oDio1.olion Value-ominaisuuden arvoksi. OOPic.Hz1 arvo vaihtuu 0:sta 1:n ja takaisin kerran sekuntissa. Kytettäessä oDio1-olion I/O-linjaan LED tämä vilkkuu jännitteen vaihtelun tahdissa kerran sekuntissa.

OOPic.Hz1-ominaisuuden tilan muutos voidaan tunnistaa ohjelmassa tapahtumana. Muokattaessa ohjelmasta tapahtumaohjattu voidaan kutsua aliohjelmaa ja siinä tehdä monimutkaisempiakin toimintoja.

### 4.5 Lisää Event (tapahtuma)-olio sovellukseen

Tapahtumaohjattujen (*Event Driven*) rutiinien lisääminen sovellukseen tapahtuu lisäämällä ohjelmaan tarvittava koodi.

Koska tämä sovellus perustuu "EkaVC" sovellukseen, joka tehtiin kappaleessa 3, otetaan sen ohjelmakoodi pohjaksi.

[Dim WIRE As New oGate](#)

[Dim LED As New oDio1](#)

```

Sub Main()
    LED.IOLine = 31
    LED.Direction = cvOutput
    WIRE.Input1.Link(OOPic.Hz1)
    WIRE.Output.Link(LED.Value)
    WIRE.Operate = cvTrue
End Sub

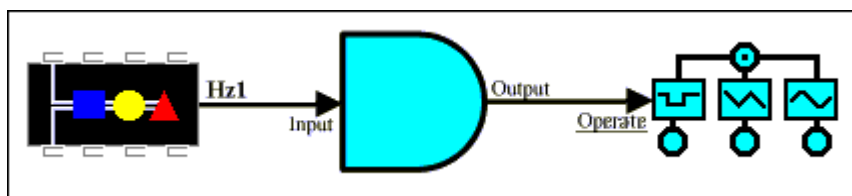
```

Kuten aiemmin mainittiin, oEvent-oliota käytetään kutsumaan monimutkaisempia rutiineja sisältävää aliohjelmaa. Lisätään edellisen koodin alkuun seuraava lause:

```
Dim BLINK As New oEvent
```

Tämä luo oEvent-oliosta ilmentymän nimeltä "BLINK". Olion toiminta on kutsua aliohjelmaa kun sen Operate-ominaisuus muuttuu 1:ksi.

Seuraavaksi virtuaalipiiriä tulee muuttaa. Kappaleessa 3 "WIRE"-olion Output-lähtö linkitettiin oDio1:n Value-ominaisuuteen. Tämä tulee linkittämään nyt oEvent-olion Operate-ominaisuuteen.



Etsi rivi, jossa lukee "WIRE.Output.Link(LED.Value)", ja muuta se muotoon:

```
WIRE.Output.Link(BLINK.Operate)
```

Tämä kytkee "WIRE"-olion Output-ominaisuuden "BLINK"-olion Operate-ominaisuuteen.

Kun ohjelma suorittaa koodirivin, virtuaalipiiri aloittaa seuraavan toiminnon ja jatkaa sitä kunnes WIRE.Operate asetetaan takaisin 0:ksi.

OOPic.Hz1-ominaisuuden arvo luetaan oGate-olioon.

oGate suorittaa loogisen OR (TAI) -operaation muiden tulojen kanssa. Tässä sovelluksessa ei ole käytössä kuin yksi tulo, joten lähdön tulos on aina sama kuin tulon arvo.

BLINK.Operate arvoksi asetetaan loogisen OR-operaation tulos.

Aina, kun BLINK.Operate ominaisuus vaihtuu 0:sta 1:ksi, "BLINK"-olion aliohjelma suoritetaan.

Seuraavassa on kokonaisuudessa koodi, johon on tehty edellä mainitut muutokset:

```
Dim BLINK As New oEvent
```

```
Dim WIRE As New oGate
Dim LED As New oDio1

Sub Main()
    LED.IOLine = 31
    LED.Direction = cvOutput
    WIRE.Input1.Link(OOPic.Hz1)
    WIRE.Output.Link(BLINK.Operate)
    WIRE.Operate = cvTrue
End Sub
```

#### **4.6 Tee Event (tapahtuma)-olion aliohjelma**

Jokaisen oEvent-olion on yhdistetty aliohjelma, joka suoritetaan, kun oEvent-olion Operate-ominaisuus asetetaan 1:ksi. Tämän aliohjelman nimi on sama kuin oEvent-olion nimi +”\_CODE”.

Lisää ohjelman loppuun seuraavat rivit:

```
Sub BLINK_CODE()
    LED.Value = 1
    LED.Value = 0
    LED.Value = 1
    LED.Value = 0
    LED.Value = 1
    LED.Value = 0
End Sub
```

Seuraava koodi sisältää täydellisen ohjelmalistauksen ohjelmasta, joka kirjoitettiin edellisissä kappaleissa.

```
Dim BLINK As New oEvent
Dim WIRE As New oGate
Dim LED As New oDio1

Sub Main()
    LED.IOLine = 31
    LED.Direction = cvOutput
    WIRE.Input1.Link(OOPic.Hz1)
    WIRE.Output.Link(BLINK.Operate)
```

```
WIRE.Operate = cvTrue  
End Sub
```

```
Sub BLINK_CODE()  
    LED.Value = 1  
    LED.Value = 0  
    LED.Value = 1  
    LED.Value = 0  
    LED.Value = 1  
    LED.Value = 0  
End Sub
```

#### **4.7 Tapahtumaohjatun sovelluksen lataaminen ja ajaminen**

Tapahtumaohjatun sovelluksen lataaminen ja ajaminen ei eroa muiden OOPic sovellusten ajamisesta.

F5-näppäimen painaminen kääntää ohjelman ja lataa "oex" tiedoston OOPiciin. Jos virheitä huomataan, kääntäminen keskeytyy ja virheet ilmoitetaan näytössä. Muutoin luodaan "oex" (OOPic Executable) tiedosto. Jos et ole tallettanut ohjelmaa aikaisemmin tai olet muuttanut sitä viimeisen talletuksen jälkeen, kääntäjä pyytää sinua tallettamaan ohjelman. Jos kyseessä on ensimmäinen talletus, kääntäjä kysyy ohjelmatiedoston nimeä. Nimeä tämä ohjelma nimelle "EkaEV".

Jotta suoritettava ohjelma voidaan siirtää OOPiciin, ohjelmointikaapelin tulee olla kytketty PC:n ja konfiguroitu oikein. Sen toisen pään tulee olla kytkettynä OOPicin ohjelmointiliittimeen ja virtalähde tulee olla kytkettynä OOPiciin. Katso kohdasta "Ohjelmointikaapeli" lisätietoja.

## 5 Ohjelmoinnin perusteet

### 5.1 OOP

OOPic Basicin syntaksi on 100% Microsoft Visual Basic yhteensopiva. Kuten Visual Basicissa, ohjelmoit OOPiciä käyttäen olio-ohjelmointia, OOP(*Object-oriented programming*).

Olio-ohjelmointi (OOP) on ohjelmointimenetelmä, joka pyrkii matkimaan tapaamme ymmärtää reaali maailman toimintaa. OOP käyttää hyväksi meidän luonnollista taipumusta yleistää ja luokitella asioita ja antaa meille mahdollisuuden yhdistellä loogisesti toisiinsa kuuluvia osioita sovelluksesta. Useimmissa OOP-kielissä kielen laajennukset sallii muodostaa erittäin monimutkaisia oliomalleja, jotka voivat matkia läheisesti reaali maailman olioita. OOPicissä suurin hyöty olio-ohjelmoinnista tulee fyysisten laitteistokomponenttien helposta yhdistettävyydestä itse OOPiciin.

Ohjelmistoteollisuus määrittää useita peruseriaatteita, jotka luokittavat kielen oliokieleksi. Näitä ovat kapselointi, periminen ja polymorfismi. Vaikka OOPicin ohjekirjoissa ei periaatteisiin useinkaan viitata, niiden tunteminen on silti hyödyllistä.

Kapselointi tarkoittaa yhteen liittyvien asioiden ryhmittelyä yhdeksi. Sen sijaan, että loogisesti yhteenliittyviä muuttujia ja funktioita käsiteltäisiin erillään, kapselointi yhdistää ne kokonaisuuksiksi, joita kutsutaan luokiksi. Muut ohjelman osat voivat käyttää näitä muuttujia ja funktioita yhtenä oliona. Kapseloinnin jälkeen muuttujia kutsutaan olion ominaisuuksiksi ja funktioita olion metodeiksi. Kapseloinnin myötä suuri osa olion monimutkaisuudesta jää olion sisäiseksi. Ainoastaan valitut metodit ja ominaisuudet tulevat julkisiksi muille osille ohjelmaa. Näitä julkisia metodeja ja ominaisuuksia kutsutaan olion rajapinnaksi. Termillä tiedon abstraktio viitataan siihen, että olion rajapinta ei paljasta olion sisäistä tapaa tallettaa tiedot ja suorittaa metodinsa. Tuloksena on rajapinta, jonka kautta voidaan käyttää monimutkaisiakin tuloja.

Polymorfismi on Kreikan kielestä lainattu termi. Se tarkoittaa kirjaimellisesti "Omata monta muotoa". Olio-ohjelmoinnissa tällä tarkoitetaan sitä, että kutsutaan oliota ilman, että tiedetään olion tarkkaa tyyppiä ja kutsuttavan metodin tekemä toiminto riippuu olion tyypistä.

Periytyemisessä voidaan luoda uusi johdettu olio, joka perii ominaisuuksia perusluokastaa. Koska OOPicin oliot perustuvat laitteiston käsittelemiseen ja rutiinit ovat kirjoitettu optimoidulla konekielellä ja talletettu PROM-muistiin, ei ole useinkaan mielekästä luoda uutta oliota, joka perii luokkia toisesta oliosta. Kuitenkin tekemällä virtuaalipiirejä voidaan periytymistä tehokkaasti toteuttaa.

### 5.2 OOPic sovelluksen rakenne

OOPic sovellukset voivat sisältää useita erilaisia komponentteja.

Main-proseduuri. Pääohjelma sisältää ohjelman kulun pääosat.

Aliohjelmat (*Sub-procedures*). Näihin voidaan kirjoittaa rutiinit, joita voidaan kutsua useista kohdista varsinaista ohjelmaa.

Oliot (*Objects*). Olioiden voidaan ajatella olevan taustalla moniajossa pyörivää koodia, jolla on omat muuttujansa. Niiden avulla kaikki OOPicin ohjelmaosiot keskustelevat laitteiston kanssa.

Tapahtumat (*Events*). Tapahtumat ovat toimintoja tai olion havaitsemia tilanteita, jotka aiheuttavat aliohjelman suorituksen vastineena olion tilan muutokseen. For more information on

Virtuaalipiirit (*Virtual Circuits*). Virtuaalipiirit emuloivat sähköpiirejä ja pyörivät taustalla. Ne voivat liittyä muuttujiin ja muuttaa minkä tahansa muuttujan arvoa.

### 5.3 Kuinka tapahtumaohjattu sovellus toimii

Tapahtuma on olion havaitsema toiminto. Tapahtumaohjattu sovellus suorittaa ohjelmakoodin, jonka ohjelmoija on kyseiselle tapahtumalle vastineeksi kirjoittanut. OOPicin oEvent-olio on ainoa olio, joka voi reagoida tapahtumiin. Jos jokin tapahtumaolion ilmentymä havaitsee signaalin käynnistää tapahtumaan liittyvän koodin suoritus, OOPic sovellus käynnistää aliohjelman, jonka nimi on ilmentymän oEvent-olion nimi + "\_CODE".

Vaikka oEvent-olio on ainoa olio, joka voi liittyä tapahtumiin, muut oliot voivat silti epäsuorasti käynnistää tapahtumaan liittyvän koodin suorituksen. Muut oliot voidaan linkittää oEvent-olioon ja saada siten liitynnän tapahtumiin. Kun haluat koodisi reagoivan tapahtumaan, sinun täytyy ensin luoda oEvent-olion ilmentymä Dim-komennolla, kirjoittaa koodi tapahtumalle ja liittää koodi tapahtumaan ja lopuksi kytkeä oEvent-olio olioon, joka käynnistää tapahtuman.

Seuraavana on esimerkki, jossa jännitteen muutos yhdessä OOPicin I/O-linjassa kutsuu käyttäen oDio1-oliota kutsumaan "MYEVENT"-aliohjelmaa.

```
Dim MyEvent As New oEvent
Dim ThingA As New oDio1
Dim Wire as New oFanout
Sub Main()
    ThingA.IOLine = 1
    ThingA.Direction = cvInput
    Wire.Input.Link(ThingA)
    Wire.Output1.Link(MyEvent.Operate)
END SUB
```

```
Sub MyEvent_Code
```

```
    'event code goes here'
```

```
End Sub
```

Suoritettuaan Main-proseduurin rivit pääohjelma jää vapaalle ja voisit lisätä ohjelmaan koodia, joka tekisi jotain muuta. Linkki tehtiin, kun oFanout-olion Input-ominaisuus linkitettiin oDio1-olioon ja Output-ominaisuus oFanout-oliosta linkitettiin Operate-ominaisuuteen oEvent-oliossa. Nyt, jos jännite I/O-linjassa muuttuu 0:sta +5 V:n ”ThingA”-nimisen olion Value-ominaisuus muuttuu 0:sta 1:ksi. Tällöin aliohjelma nimeltä ”MyEvent\_Code” suoritetaan. Kun End Sub-lauseke saavutetaan aliohjelman lopussa, ohjelman suoritus jatkuu siitä, missä se oli tapahtuman sattuessa.

### 5.3.1 Tapahtumaohjattu vs. Perinteinen ohjelmointi

Perinteisessä tai proseduraalisessa sovelluksessa sovellus itse määrittää mikä osa koodista suoritetaan seuraavaksi. Suoritus käynnistyy ensimmäiseltä ohjelmariviltä ja seuraa ohjelmavuon kulkua tarvittavine haarautumisineen ja aliohjelmakutsuineen.

Verrattuna yllä olevaan esimerkkiin perinteisen ohjelman täytyy koko ajan tarkkailla I/O-linjan tilaa määritelläkseen milloin haaraudutaan ”MYEVENT\_CODE”-aliohjelmaan. Tämä vie prosessointiaikaa sovellukselta ja voi monimutkaistaa ohjelman rakennetta.

Tapahtumaohjatussa ohjelmassa muutos laitteiston tilassa voi aiheuttaa aliohjelmakutsun vaikka ohjelmassa ei siihen olisi varauduttukaan. Tällöin se järjestys, jossa ohjelmaa suoritetaan, riippuu siitä mitä laitteistossa tapahtuu. Tämä on tapahtumaohjatun ohjelmoinnin tärkein puoli: ohjelma reagoi silloin kun tarvitaan.

Koska ohjelman suoritusjärjestys ei määritä tapahtuman havaintohetkeä, ohjelmakoodin täytyy tehdä oletuksia ympäröivän maailman tilasta suoritushetkellä. Ohjelmakoodi voi käynnistää muita tapahtumia reagoidessaan tapahtumaan, ohjelmakoodia kirjoitettaessa täytyy tällöin seurannaisvaikutukset huomioida.

Huom! Vältä liian monien tapahtumien yhtäaikaista suorittamista. Tämä voi aiheuttaa pinon ylivuodon.

### 5.3.2 Ohjelmakoodi, joka suoritetaan OOPicin käynnistyessä

OOPic Basicin syntaksissa ”MAIN”-niminen aliohjelma on määritetty ohjelman suorituksen aloituskohdaksi. Kun OOPiciin kytketään käyttäjännite tai se resetoidaan, ohjelman suoritus alkaa ”MAIN”-aliohjelman ensimmäiseltä riviltä. Jos haluat lopettaa ohjelman suorituksen ja odottaa jotain tapahtumaa, tule yksinkertaisesti vain ulos ”MAIN”-aliohjelmasta. Toinen tapa pysäyttää ohjelman suoritus on asettaa OOPic-olion Operate-ominaisuuden arvoksi 0 (cvOff). Tällöin OOPic sammuttaa

kaikki tapahtumaa odottavat rutiinit ja menee vähän virtaa kuluttavaan valmiustilaan.

Ennen kuin "MAIN"-aliohjelman suoritus käynnistyy, kun OOPiciin kytketään käyttöjännitte tai se resetoidaan, kaikki sovelluksen oliot luodaan ja ne alustetaan siten, että "MAIN"-aliohjelmalla on mahdollisuus käsitellä niitä.

### 5.3.3 Tapahtumaohjatun ohjelman lopetus

Tapahtumaohjatun OOPic-sovelluksen suoritus loppuu, kun kaikki oEvent-oliot ovat irrotettu irti linkeistään ja viimeisen käynnissä olevan rutiinin suoritus loppuu. Koska mitään koodia ei ole käynnissä, mitään aliohjelmaa käynnistävää tapahtumaa ei voi tulla ja ei ole olemassa mitään keinoa ohjelman uudelleenkäynnistymiseksi.

Vaikka mitään ohjelmakoodia ei ole käynnissä, oliot toimivat tausta-ajossa kunnes virrat katkaistaan. Koska Operate-ominaisuus voidaan asettaa 0:ksi toisesta oliosta tai vaikka tietokoneesta käsin I2C-väylän kautta, ohjelmakoodin suorituksen ei tarvitse olla käynnissä OOPicin sammuttamiseksi.

## 5.4 Oliot

OOPicin ohjelmointikieliet hyväkseen olioita (Objects). Oliot ovat kokoelma ohjelmakoodia ja tietoa, jotka toimivat yhtenä kokonaisuutena. Kappaleesta 6 löytyy yksityiskohtaisempia tietoja olioista. OOPicissä on valmiina useita olioluokkia käytettäväksi hyväksi tehtäessä omia sovelluksia.

Useimmissa Basic-kielissä on olemassa useita lauseke ja funktioita, jotka käsittelevät samaa aihetta ja liittyvät täten toisiinsa. Esimerkiksi Peek funktio ja Poke lauseke käsittelevät molemmat muistia. Peek funktio palauttaa arvon annetusta muistipaikasta ja Poke lauseke asettaa annetun arvon haluttuun muistipaikkaan, kuten alla olevasta esimerkistä voidaan havaita.

```
V = Peek(128) `Vanha tapa lukea muistipaikan 128 arvo  
muuttujaan V.
```

```
Poke 128,V `Vanha tapa kirjoittaa muuttujan V arvo  
muistipaikkaan 128.
```

Näiden kahden lauseen syntaksit ovat täysin erilaiset. OOPic Basicissä nämä kaksi operaatiota on yhdistetty yhteen olioon nimeltään "oRAM". oRAM oliolla määritettyä muistipaikkaa käsitellään yhtenä yksikkönä ja sitä käytetään kuin muuttujaa. Seuraava esimerkki kuvaa oRAM olion nimeltä "MYRAM" käyttöä.

```
V = MYRAM `Uusi tapa lukea muistipaikan sisältö  
muuttujaan V.
```

```
MYRAM = V `Uusi tapa kirjoittaa muuttujan V arvo  
muistipaikkaan.
```

Tämä koodi on helpommin luettavaa, yhtenäisempää ja helpommin ymmärrettävää. Tämä on yksi olio-ohjelmoinnin eduista.

OOPic sovelluksissa olio määritellään (tai sen ilmentymä luodaan) Dim lausekkeella seuraavasti:

```
Dim Object As New type 'Dim lausekkeen syntaksi
```

Jokaisella oliolla on oltava yksilöllinen ohjelmoijan määrittämä nimi. Olion nimessä ei isoilla ja pienillä kirjaimilla ole merkitystä ja se toimii tunnisteiden läpi ohjelman kyseiseen olioon. Sama nimeämiskäytäntö pätee kaikkiin muihinkin tunnisteisiin OOPic-kielessä.

## 5.5 Muuttujat

Ohjelmia kirjoitettaessa tulee usein tarve tallettaa jonkin laskutoimituksen tulos väliaikaisesti jonnekin. Esimerkiksi vertailtaessa usean laskutoimituksen tuloksia keskenään pitää arvot tallettaa johonkin. Muuttujia käytetään arvojen varastointipaikkana.

OOPicin kielissä, kuten useimmissa ohjelmointikielissä, käytetään muuttujia arvojen talletuspaikkoina. Muuttujilla on nimi, joiden avulla niihin voidaan viitata ja niillä on arvo, joka niihin on sijoitettu. Toisin kuin yleensä, OOPicin muuttujat ovat myös olioita. Tämä antaa OOPic-sovelluksille joitain etuja. Tärkein etu on, että muuttujilla on ominaisuuksia, joiden avulla voidaan hallita muuttujan käytöstä. Toinen tärkeä ominaisuus on, että koska muuttujan arvo (*Value*) on olion ominaisuus, se voidaan linkittää virtuaaliin ja siihen voidaan päästä käsiksi muista olioista ja muista tietokoneista/mikrokontrollereista I2C-verkon kautta.

OOPic Basicin syntaksissa muuttuja määritellään (tai sen ilmentymä luodaan) Dim-lausekkeella seuraavasti:

```
Dim variable As New type 'Dim lausekkeen syntaksi
```

Huomaa, että samaa syntaksia käytetään luotaessa minkä tahansa tyyppinen olio.

Jokaisella muuttujalla on oltava yksilöllinen ohjelmoijan määrittämä nimi. Olion nimessä ei isoilla ja pienillä kirjaimilla ole merkitystä ja se toimii tunnisteiden läpi ohjelman kyseiseen muuttujaan. Sama nimeämiskäytäntö pätee kaikkiin muihinkin tunnisteisiin OOPic-kielessä.

## 5.6 Aliohjelmat

Ohjelmoija voi yksinkertaista ohjelmointitehtävää jakamalla ohjelman pienempiin loogisiin osiin, joita kutsutaan aliohjelmiiksi. Aliohjelma toteuttaa yhden yksittäisen toiminnon. Sitä käytetään, kun samaa toimintoa halutaan kutsua useita kertoja eri puolilta ohjelmaa. Yleisesti ottaen aliohjelma voi tehdä kaiken sen minkä pääohjelmakin ja lisäksi palata pääohjelmaan kohtaan, josta sitä kutsuttiin. Aliohjelmien käyttö säästää ohjelmamuistia, ohjelman kirjoitusaikaa ja tekee ohjelmista helpompia lukea ja debugata.

Call-komentoa käytetään aliohjelman kutsumiseen. Viimeinen rivi aliohjelmassa on End Sub. End Sub lauseke aiheuttaa paluun Call-lauseketta seuraavalle riville.

Jokaisella aliohjelmalla on uniikki nimi, jonka ohjelmoija määrittää. Aliohjelman nimessä ei isoilla ja pienillä kirjaimilla ole eroa (*case insensitive*) ja se toimii tunnisteena, jonka avulla aliohjelmaan viitataan. Ainoa vaadittava aliohjelma on "MAIN", joka on aliohjelma, jota kutsutaan kun OOPic käynnistyy tai resetoidaan.

Kaikki muuttujat OOPic Basicissa ovat globaaleja. Tämä tarkoittaa, että mikä tahansa aliohjelma voi muokata mitä tahansa muuttujaa, joka voi sijaita missä tahansa kohti ohjelmaa.

## 5.7 Funktiot

OOPicin kielet sisältävät järjestelmän tarjoamia tai luontaisia funktioita kuten Sin. Funktio on proseduuri, joka saa argumentteja, suorittaa laskentaa tai operaatioita ja palauttaa arvon. Funktiota käytettäessä proseduurin nimi ja argumentit kirjoitetaan ohjelmakoodiin. Esimerkiksi voit käyttää Sin-funktiota seuraavasti saadaksesi kulman sinin arvon:

```
newvalue = Sin(oldvalue)
```

## 5.8 Ohjausrakenteet

Ohjausrakenteet ohjelman suorituksen kulun ohjaamisen. Mikäli suorituksen kulkuun ei mitenkään puututa, ohjelma käy läpi järjestyksessä lausekkeet kunnes saapuu ohjelman loppuun. Erittäin yksinkertaisissa tapauksissa tämä on riittävää, mutta ohjelmoitavuuden hyöty tulee yleensä siitä, että käskyjen suoritusjärjestyksestä voidaan muuttaa erilaisten rakenteiden ja silmukoiden avulla tilanteen mukaan.

### 5.8.1 Vuonohjaus

Vuonohjauslausekkeet mahdollistavat muutoksen ohjelman suoritusjärjestykseen.

OOPic Basicin syntaksin vuonohjausrakenteet ovat:

```
Goto
```

```
Call
```

### 5.8.2 Valintarakenteet

Valintarakenteet ovat lausekkeitä, joiden avulla voidaan testata ehtoja ja riippuen testauksen tuloksesta valita haluttu operaatio. Ehto on yleensä vertailu, mutta se voi olla mikä tahansa lauseke, jolla on numeerinen arvo. Numeerinen arvo tulkitaan siten, että 0 on epätosi (*False*) ja mikä tahansa muu arvo on tosi (*True*). Vakioita cvTrue ja cvFalse voidaan käyttää ohjelmassa 0 ja 1 tilalla.

OOPic Basicin syntaksin tukemat valintarakenteet ovat:

[If...Then...Else](#)

[Select Case](#)

### 5.8.3 Silmukkarakenteet

Silmukkarakenteet mahdollistavat yhden tai useamman ohjelmariivin toiston useampaan kertaan.

OOPic Basicin syntaksin tukemat silmukkarakenteet ovat:

[For...Next](#)

[Do...Loop](#)

## 5.9 *Kommenttien lisääminen ohjelmakoodiin*

Koodia kirjoitettaessa on hyödyllistä lisätä koodin joukkoon kommentteja ajatellusta ohjelman toiminnasta. Rem -lauseke kertoo sovellukselle, että sitä seuraavien sanojen kohdalla ei tehdä mitään. Kommentti voi olla lausekkeen perässä tai se voi olla ainoa asia koko rivillä. Kommentti voidaan aloittaa myös '-merkillä.

## 6 Johdanto olio-ohjelmointiin

### 6.1 Mitä on OO?

OO on akronyymi sanoista olio-ohjelmointi (Object-Oriented Programming). Termillä tarkoitetaan ohjelmointikieliä, joiden käyttö pohjautuu olioihin.

Olio-ohjelmointi on yritys tehdä tietokoneohjelmien kirjoittamisesta helpompaa käyttäen esimerkkinä reaali maailman fyysisiä objekteja.

Teknisesti oliot OO-kielessä ovat yhdistelmä tietoa ja sitä käsittelevää ohjelmakoodia, joita käsitellään yhtenä kokonaisuutena. Jokaisella oliolla on yksilöllinen nimi ja kaikki viittaukset olioon tehdään nimen avulla.

OOPicissä kaikki laitteistotason I/O:t ja virtuaalipiirit on toteutettu olioiden avulla, jopa muuttujat ovat olioita.

### 6.2 Miksi olioita?

Olio-ohjelmoinnin hyöty tulee siitä, että ihmiset luonnostaan osaavat toimia fyysisten objektien kanssa. Kun ohjelmointikielessä ohjelmoija voi toimia cyber-objektien kanssa samalla tavalla, ohjelmoinnin oppiminen on helpompaa.

Useimmissa ohjelmointikielissä ohjelman toiminta muodostuu useista kymmenistä komennoista ja käskyistä, joiden rakenteet pitää muistaa ulkoa. Useat komennoista ovat käytössä vain kyseisessä ohjelmointikielessä. Tällöin on vaikeaa hallita kaikkien kielten eri vivahteita.

Olio-ohjelmointikielissä kaikki toiminnallisuus on siirretty olioihin. Tällöin kielen käskymäärää voidaan rajoittaa vain niihin komentoihin, jotka ovat tarpeellisia ohjelman kulun määrittelyyn ja olioiden käsittelyyn. Tämä vähentää ulkoa opeteltavien käskyjen määrän minimiin.

Perinteisissä ohjelmointikielissä ohjelman kulku etenee riveittäin läpi sovelluksen. Siirrettäessä käsittelyt olioille sovelluksesta on tullut moniajosovellus, jossa kukin olio käsittelee oman prosessinsa ja sovellus luo ja hallitsee olioita.

OOPicissä oliot ovat sovelluksen tapa kommunikoida ympäristönsä kanssa. Luotaessa OOPic-sovellusta ohjelmoijan tarvitsee muistaa ulkoa pieni joukko vuonohjauskomentoja ja kuinka sovelluksessa käytettäviä oliota kutsutaan.

### 6.3 Mitä olioilla voi tehdä?

Koska OOPicin oliot ovat suoraan yhteydessä PIC-mikrokontrollerin laitteiston eri osiin, sovellus voi toimia nopeasti. Samalla laitteistoa voidaan käsitellä korkean tason ohjelmointikielellä sen sijaan, että käytettäisiin muistiosoiteviittauksia ja bittioperaatioita laitteiston käsittelyyn.

Olio sisältää myös moniajokoodia, joka toimii yhdessä sovelluksen, laitteiston ja muiden olioiden kanssa. Tämä antaa ohjelmoijalle käyttöön valmiiksi kirjoitettuja ohjelmakirjastoja. Jos esimerkiksi käytetään oA2D

oliota lukemaan OOPicin I/O-tulossa olevaa jännitettä, luotaessa olion ilmentymä ja määrittelemällä sen ominaisuudet, sovellus saa lukuarvon, joka vastaa I/O-linjalla olevaa jännitettä. Ohjelmoijan ei tarvitse huolehtia kuinka varsinainen muunnos analogisesta jännitteestä digitaaliseksi suoritetaan, koska olion moniajossa toimiva koodi huolehtii siitä.

Koska oliot suorittavat toimintonsa moniajossa, OOPic sovellus voi olla yhteydessä useaan laitteiston osaan samanaikaisesti. Tämä tarkoittaa, että sovelluksen ei tarvitse pysäyttää menossa olevaa työtään tehdäkseen jotain muuta.

Linkittämällä olioita yhteen voidaan sovellukseen luoda virtuaalipiiri. Virtuaalipiiri on OOPicin toiminto, joka näyttää toiminnallisesti diskreetiltä elektroniikkapiiriltä, mutta on itse asiassa joukko OOPicin olioita, joka emuloi piirin toimintaa. Virtuaalipiirit tehdään ohjelmallisesti yhdistämällä toisiinsa OOPic oliot samalla tavalla kuin tehtäisiin yhdistettäessä toisiinsa fyysisesti elektroniikkakomponentteja. Jokaista yksittäistä muodostetun virtuaalipiirin osaa voidaan muokata tai sen arvoa käyttää missä tahansa ohjelman osassa.

Seuraavassa taulukossa on kuvattu OOPicin kielten erilaiset oliotyypit.

Tyyppi	Kuvaus
Laitteisto ( <i>Hardware</i> )	Olio, joka edustaa fyysisesti olemassa olevaa laitteiston osaa ja/tai kapseloi sen OOPiciin kieleen. Sovelluksessa voi olla useita laitteisto-olion ilmentymiä määritettynä, mutta vain yksi niistä voi olla käytössä kutakin laitteiston osaa kohti samanaikaisesti.
Prosessointi ( <i>Processing</i> )	Olio, joka saa arvonsa toisilta olioilta, suorittaa määritetyt laskumat ja tallettaa tuloksen toiseen olioon. Nämä oliot toimivat virtuaalipiirien rakennusosasina. Prosessointiolioista voidaan määrittää niin monta ilmentymää kuin muistiin mahtuu.
Muuttuja ( <i>Variable</i> )	Olio, joka varastoi arvon ja jonka avulla kyseinen arvo voidaan lukea. Muuttujaoliosta voidaan määrittää niin monta ilmentymää kuin muistiin mahtuu.
Järjestelmä ( <i>System</i> )	Olio, joka kontrolloi useita järjestelmätoimintoja. Järjestelmäolio on aina läsnä, kun OOPic on käynnissä. Järjestelmäoliosta ei voi määrittellä uusia ilmentymiä.

## 6.4 Mistä oliot tulevat?

Yksittäinen lauseke sovelluksen koodissa luo olion. OOPicin Basicin syntaksissa ohjelmoija käyttää *Dim* lauseketta olion luomiseen. *Dim* lausekkeessa ohjelmoijan tulee määrittää yksilöllinen nimi uudelle oliolle ja se, minkä tyyppinen olio luodaan. Kun sovellusta suoritetaan ja saavutaan kyseiseen lausekkeeseen, luodaan ko. tyyppin olion ilmentymä.

Seuraavassa esimerkissä on lauseke, jolla luodaan oByte-tyyppisen olion ilmentymä nimeltä "XYZ".

```
Dim XYZ As New oByte
```

OOPiciissä on useita erilaisia oliotyyppisiä, joita sovellus voi luoda. Jokaiselle näille eri tyypeille on olemassa oma olioluokka. Olioluokka määrittää kuinka kukin olio toimii. Jotta olion ja sen luokan välinen yhteys kävisi helpommin selville, ajatellaan esimerkkinä autoa ja sen piirustuksia. Piirustukset vastaavat luokkaa, koska ne määräävät millainen autosta tulee, esim. koko, muoto ja kuinka auto toimii. Piirustusten mukaan voidaan tehdä yksi tai useampia autoja ja autot, jotka tehdään ovat olioita. Jokainen tehty olio on kyseisen luokan ilmentymä.

Kun olio luodaan, se on identtinen kopio luokastaan. Kun se on olemassa yksilöitynä oliona, sitä voidaan muokata muuttamalla sen ominaisuuksien arvoja. Esimerkiksi, jos luot 3 oliota käyttäen samaa luokkaa, jokainen näistä 3 oliosta ovat luokan identtisiä ilmentymiä. Jokaisella 3 uudella oliolla on samat piirteet ja kyvyt (omianisuudet, metodit ja tapahtumat), jotka luokkaa määrittää. Jokaiselle täytyy kuitenkin antaa erilainen nimi ja luonnin jälkeen kaikkien olioiden ominaisuuksia ja metodeja voidaan erikseen kontrolloida.

Kaikkien Dim-lausekkeiden tulee olla sovelluksen alussa. Kun OOPic käynnistetään ensimmäistä kertaa tai se resetoidaan, Dim lausekkeet suoritetaan ensimmäisinä ja täten kaikki sovelluksen oliot luodaan ja niiden arvot alustetaan ennen muun ohjelmakoodin suorittamista.

## 6.5 Olioiden luominen

OOPicin Basicin syntaksissa sovelluksen täytyy ilmoittaa aikeestaan luoda olion ilmentymä Dim lausekkeella. Tämä lauseke varaa tarvittavan muistitilan vapaalta RAM-muistin alueelta olion ilmentymälle. Ohjelmoija antaa ilmentymälle nimen, jolla sovellus tunnistaa sen. Mikäli yritetään viitata olioon, jota ei ole vielä luotu, aiheutuu käännösvaiheessa virhe ja käännös keskeytyy.

Seuraavassa esimerkissä tehdään ilmentymä oliosta, jonka nimeksi tulee "TheNumberThatWeWant" ja olion tyyppiä tulee oByte.

```
Dim TheNumberThatWeWant As New oByte
```

### 6.5.1 Olioiden nimet

Jokaisella oliolla sovellusohjelmassa täytyy olla yksilöllinen nimi jotta sovellus voi viitata siihen. Olioiden nimet voivat olla mitä tahansa kunhan ne noudattavat määritettyä nimeämiskäytäntöä. Seuraavassa on nimeämiskäytännön säännöt:

- Olion nimen pitää alkaa kirjaimella
- Olion nimessä ei saa olla pistettä.

- Olion nimi ei saa olla yli 32 merkkiä pitkä.
- Olioiden nimien tulee olla yksilöllisiä koko sovelluksen sisällä. (Isoilla ja pienillä kirjaimilla ei ole merkitystä)

Olion nimeä ei talleteta RAM-muistiin, joten nimet voi olla pitkiä tahansa (32 merkkiin asti) ilman, että sillä olisi vaikutusta olion ilmentymälle varattavan muistin määrään.

Olioiden nimissä isoilla ja pienillä kirjaimilla ei ole merkitystä eli nimet "ThingOne" ja "thingone" viittaavat samaan olioon.

### 6.5.2 Taulukoiden luominen olioista

Olioista voidaan luoda taulukko määrittämällä taulukon koko Dim-lausekkeessa. Seuraavassa esimerkissä on luotu kaksi oliotaulukkoa, ensin 3 oByte-olion taulukko ja toisena 6 oDio1-olion taulukko.

```
Dim BA(3) As New oByte
```

```
Dim PA(6) As New oDio1
```

Näihin olioihin voidaan nyt viitata käyttäen apuna indeksinumeroa. Seuraavassa esimerkissä kaikki arvot BA(1):sta BA(6):n on laskettu yhteen ja sijoitettu Z:n.

```
For X = 1 to 6
    Z = Z + BA(X).Value
Next X
```

### 6.5.3 Olioiden luominen luokista, joiden koko on muuttuva

Kolmen olion, oBuffer, oGate ja oFanout, luokkamäärittelyssä on muuttuva luokan koko.

oBuffer-olion yhteydessä koko määrittää kuinka monta tavua olion puskuri on.

Seuraavassa esimerkissä oBuffer-oliosta tehdään ilmentymä, jonka koko on 8 tavua.

```
Dim bunchobytes As New oBuffer(8)
```

oGate-olion yhteydessä koko määrittää kuinka monta tuloporttia portissa on.

Seuraavassa esimerkissä oGate-oliosta tehdään ilmentymä, jossa on kaksi tuloa.

```
Dim thegate As New oGate(2)
```

oFanout-olion yhteydessä koko määrittää kuinka monta lähtöä oliossa on.

Seuraavassa esimerkissä oFanout-oliosta tehdään ilmentymä, jossa on kaksi tuloa.

```
Dim somewires As New oFanOut(2)
```

Lista kaikista OOPicin oliosta löytyy verkosta osoitteesta [www.oopic.com](http://www.oopic.com).

## 6.6 Olioiden kanssa työskentely

OOPicin oliot tukevat ominaisuuksia, metodeja ja tapahtumia. Olion käyttäytymistä voi muuttaa muuttamalla sen ominaisuuksia. Ominaisuuksien lisäksi oliolla on metodeja. Menetrit ovat osa oliota kuten ominaisuudetkin. Yleisesti ottaen menetrit ovat toimintoja, jotka suoritetaan oliolla kun taas ominaisuudet ovat arvoja, joita asetetaan ja luetaan. Olioihin liittyy myös tapahtumat. Tapahtumat liipaistaan, kun jokin oliossa muuttuu ja sen seurauksena suoritetaan jokin ohjelmakoodin pätkä.

Kun olion ilmentymä on luotu, kaikki viittaukset ko. ilmentymään tehdään kirjoittamalla ilmentymän nimi, piste ja halutun ominaisuuden tai metodin nimi, johon halutaan viitata.

## 6.7 Olion ominaisuuksien käsittely

Olion ominaisuudet ovat arvoja, joita se pitää tallessa. Nämä arvot voivat olla numeerisia arvoja, joita pidetään tallessa tai jotka määrittää, mitä tehdään tai jotka ilmaisevat, että jotakin on tehty.

### 6.7.1 Ominaisuuden arvon asettaminen

Ominaisuuden arvo asetetaan, kun halutaan muuttaa olion käyttäytymistä. Esimerkiksi, oDio1-olion Direction ominaisuuden arvo asetetaan sen mukaan halutaanko ko. I/O-linjan olla tulo vai lähtö.

Olion ominaisuuden arvo asetetaan määrittämällä olion nimi, piste ja asetettava ominaisuus.

Seuraavassa on olion ominaisuuden asettamisen syntaksi:

```
Object.property = expression
```

Seuraavassa esimerkissä demonstroidaan ominaisuuksien asettamista:

```
Dim ThingA As New oDio1
```

```
Sub Main()
```

```
    ThingA.IOLine = 1
```

```
    ThingA.Direction = cvInput
```

```
End Sub
```

Tässä esimerkissä olion nimeltä ThingA IOLine-ominaisuus on asetettu arvoon 1 ja Direction-ominaisuus on asetettu arvoon 1, (cvInput). cvInput on vakio, jonka arvo on 1. Niinpä sen jälkeen, kun esimerkki on suoritettu, Direction-ominaisuus oliolla nimeltä "ThingA" on 1. Kun Direction-ominaisuus oDio1-tyyppisellä oliolla on 1, olion Value-ominaisuus päivittyy I/O-linja 1:n sähköisellä tilalla.

Joillain olioilla on ominaisuuksia, jotka ovat vain luku-tyyppiä. Nämä oliot yleensä tekevät jotain laskelmia saadakseen ominaisuudelle arvon ja kirjoittaminen niihin ainoastaan hävittää nämä arvot.

### 6.7.2 Ominaisuuksien arvojen lukeminen

Ominaisuuden arvon lukeminen tapahtuu, kun sovelluksen tarvitsee saada tietoa olion tilasta. Esimerkiksi ohjelma voi päätellä onko I/O-linjalla jännite lukemalla Value-ominaisuuden arvon oDio1-oliosta, joka osoittaa kyseiseen I/O-linjaan.

Olion ominaisuuden arvo luetaan määrittämällä olion nimi, piste ja luettava ominaisuus.

Seuraavassa on olion ominaisuuden lukemisen syntaksi:

```
Variable = Object.property
```

Seuraavassa esimerkissä oDio1-olion arvo luetaan ja se asetetaan toisen oDio1-olion arvoksi.

```
Dim ThingA As New oDio1
Dim ThingB As New oDio1

Sub Main()
    ThingA.IOLine = 1
    ThingA.Direction = cvInput
    ThingB.IOLine = 2
    ThingB.Direction = cvOutput
    Do
        ThingB.Value = ThingA.Value
    Loop
End Sub
```

Ominaisuuden arvoa voidaan käyttää myös monimutkaisemmissa lausekkeissa.

Seuraavassa esimerkissä MyNumber asetetaan joko 0:ksi tai 5:ksi riippuen onko I/O-linjalla jännite lausekkeen suorittamisen aikaan.

```
Dim ThingA As New oDio1
Dim MyNumber As New oByte

Sub Main()
    ThingA.IOLine = 1
```

```
ThingA.Direction = cvInput
MyNumber.Value = ThingA.Value * 5
End Sub
```

### 6.7.3 Oletusominaisuuksien käyttö

Monilla olioilla on oletusominaisuuksia. Oletusominaisuuksia voidaan käyttää ohjelmakoodin yksinkertaistamiseen, koska koodissa ei tarvitse olla viittausta siihen asetettaessa tai luettaessa arvoa. Oletusominaisuus voidaan asettaa samalla kun olio luodaan.

Oliolle, jolle Value on oletusominaisuus, seuraavat kaksi koodiriviä ovat keskenään ekvivalentteja:

```
Object = 20
```

Ja

```
Object.Value = 20
```

Muuttuja- ja laitteisto-olioilla on Value-ominaisuus oletusominaisuutena. Kaikilla olioilla ei ole oletusominaisuutta vaan näille joudutaan aina määrittämään käytettävä ominaisuus.

Seuraavissa kahdessa koodin pätkässä muuttujan Q arvo kasvatetaan yhdellä.

```
Q = Q + 1
```

Ja

```
Q.Value = Q.Value + 1
```

Oliot, joita käytetään virtuaalipiireissä ja ovat linkitetty toisiin olioihin, käyttävät kohdeolioiden oletusominaisuuksia paitsi jos ne erityisesti linkitetty kohdeolion johonkin lippu-tyyppiseen ominaisuuteen.

## 6.8 Toimintojen suorittaminen metodien avulla

Metodi on toiminta, jonka olio voi suorittaa. Kutsuttaessa ne saavat olion suorittamaan määrätyn toiminnon.

### 6.8.1 Metodien käyttö ohjelmakoodissa

Metodia kutsutaan kun halutaan olion tekevän halutun toiminnon. Olio esimerkiksi kasvattaa Value-ominaisuutensa arvoa yhdellä, kun Inc-metodia kutsutaan.

Kutsuttaessa olion metodia koodiin kirjoitetaan olion nimi, piste ja kutsuttavan metodin nimi.

Seuraavassa on metodikutsun syntaksi.

```
Olio.metodi
```

Seuraavassa esimerkissä kutsutaa metodia:

```
Dim ThingA As New oByte
```

```
Sub Main()  
    ThingA = 1  
    ThingA.Clear  
End Sub
```

Tässä esimerkissä oliion nimeltään ThingA Value-ominaisuus asetetaan 1:ksi. Sen jälkeen kutsutaan Clear-metodia, joka asettaa Value-ominaisuuden 0:ksi.

Refer to the OOPic technical guide for a list of Object methods.

## 6.9 Tapahtumien käsittely

Tapahtuma on mikä tahansa toiminto, joka on tapahtunut oliossa. Sovellusohjelman täytyy usein käsitellä tapahtuma suorittamalla kyseiseen tapahtumaan liittyvä ohjelmakoodi.

### 6.9.1 oEvent-olio

oEvent-olio on siinä mielessä poikkeuksellinen olio, että siihen liittyvä aliohjelma suoritetaan, kun se Operate-ominaisuus asetetaan 1:ksi. oEvent-olion Operate-ominaisuus voidaan linkittää virtuaalipiiriin mahdollistaen, että mikä tahansa tapahtuma OOPicissä voi liipaista aliohjelman suorituksen.

## 6.10 Olioiden yhteen linkittäminen

Linkit OOPicin olioiden välillä antavat ohjelmoijalle mahdollisuuden luoda virtuaalipiirejä.

Useat OOPicin oliot pystyvät käsittelemään toisten olioiden ominaisuuksia. Tämä mahdollistaa virtuaalipiirien tehokkaan toiminnan. Virtuaalipiirit voivat vaihtaa tietoja ja tehdä tarvittavia laskutoimituksia taustalla ja varsinainen ohjelma voi keskittyä virtuaalipiirien tulosten käsittelyyn. Esimerkiksi ”Askelmoottorin ohjaus”-sovelluksessa virtuaalipiiri voidaan määrittää siten, että se paikoittaa askelmoottorin tarkasti paikalleen. Yksi muuttuja sisältää tiedon paikasta ja kun se muuttuu, askelmoottori kääntyy uuteen asemaansa. Kaikki, mitä varsinaisen ohjelman tarvitse tehdä, on sijoittaa muuttujaan uutta paikkaa vastaava arvo. Seuraavassa on varsinaisen ohjelman sijoituslauseen esimerkki:

```
MotorLocation = 5000
```

Virtuaalipiirin rakennuspalikoita ovat käsittelyoliot (*Processing objects*). Nämä oliot saavat tuloarvonsa ja välittävät lähtöarvonsa niihin olioiden ominaisuuksiin, joihin ne on linkitetty. Oliot linkitetään osoittimien avulla. Osoittimet ovat tietotyyppisiä, jotka kertovat oliolle, mistä toinen olio löytyy.

Osoittimia on kahdenlaisia:

- Osoitin kohdeolioon.
- Osoitin kohdeolion lippu (*Flag*)-ominaisuuteen.

Osoitin kohdeolioon kertoo linkitetulle oliolle missä kohdeolio sijaitsee. Tämän tyyppinen osoitin osoittaa olioon itseensä. Oliolinkki, joka käyttää tällaista osoitinta, asettaa tai palauttaa kohdeolion Value-ominaisuuden arvon.

Osoitin kohdeolion lippu-ominaisuuteen kertoo linkitetulle oliolle missä kohdeolion tietty lippu sijaitsee. Tämän tyyppisen oliion tulee osoittaa lippu-ominaisuuteen ja oliolinkki, joka käyttää tällaista osoitinta, asettaa tai palauttaa kohdeolion sen lippu-ominaisuuden arvon, johon linkki osoittaa.

Linkki muodostuu, kun käsittelyolion osoitin linkitetään kohdeolioon tai kohdeolion lippu-ominaisuuteen.

Seuraavassa esimerkissä kaksi oGate-olion tuloa linkitetään kahteen oDio1-olioon.

```
Dim aGate As New oGate(2)
```

```
Dim DioA As New oDio1
```

```
Dim DioB As New oDio1
```

```
Sub Main()
```

```
    aGate.Input1.Link(DioA.Value)
```

```
    aGate.Input2.Link(DioB.Value)
```

```
End Sub
```

Osoitinten ja olioiden tyyppin täytyy olla yhteensopivia. Edellisessä esimerkissä oGate-olion tulojen täytyy linkittyä lippu-tyyppisiin ominaisuuksiin ja Value-ominaisuus oDio1-oliolla on lippu-tyyppinen ominaisuus.

### **6.11 Dynaaminen tiedonsiirto (*Dynamic Data Exchange*)**

OOPicin ohjelmointikieli mahdollistaa kommunikoinnin ja tiedonvaihdon usean erillisen OOPicin välillä. Tästä on hyötyä, kun yksi OOPic keskittyy tietyn ohjauksen hoitamiseen ja halutaan hallita tai seurata sen tekemisiä toisesta OOPicistä tai tietokoneesta käsin. OOPic käyttää Philipsin I2C kaksijohdinväylää tiedon välitykseen verkkoon kytkettyjen laitteiden välillä. Enimmillään 127 eri laitetta voi olla kytkettynä samaan verkkoon käyttäen 2 johdinta ja maajohdinta. I2C-verkko ei vaadi mitään erityiskaapeleita tai komponentteja.

Lisätietoja Philipsin I2C verkosta löytyy linkistä Philipsin sivuille osoitteesta <http://www.oopic.com>

## 7 Muuttujat ja vakiot

### 7.1 Mitä ovat muuttujat?

Kuten kappaleessa 5, "Ohjelmoinnin perusteet", kerrottiin, muuttujia käytetään arvojen tallettamiseen. Useimmissa ohjelmointikielissä muuttujat ovat vain viittaus muistialueeseen, jossa kyseinen arvo sijaitsee. OOPic:n kielissä muuttujat ovat kuitenkin oliioita, jotka sisältävät arvon. Tällöin jokaisella muuttujalla voi olla muitakin ominaisuuksia kuin pelkkä arvo. Tästä on tiettyjä etuja OOPic-sovelluksissa. Tärkein etu on ominaisuuksien ja metodien olemassaolo, joilla voidaan kontrolloida muuttujan käyttäytymistä. Toinen etu on, että koska muuttujan arvo on ominaisuus, toiset oliot ja I2C-verkon mikrokontrollerit ja tietokoneet voivat käsitellä sitä. (Kts. kappaleesta 6, "Johdanto olio-ohjelmointiin" lisätietoja kuinka luodaan muuttuja-olioita ja käytetään niiden ominaisuuksia ja metodeja.)

OOPicissä on useita erikokoisia ja -tyyppisiä muuttujia erilaisiin tarpeisiin.

- oBit, oNibble, oByte ja oWord
- oBuffer
- oEEProm
- oRam

### 7.2 oBit, oNibble, oByte ja oWord muuttujat

oBit, oNibble, oByte ja oWord muuttujia käytetään arvojen tallettamiseen. Kaikki 4 muuttujatyyppeä toimivat lähes samalla tavalla, mutta lukualueen koko on erilainen.

Tyyppi	Koko	Bittien määrä	Lukualue	Oliomuistin tarve
oBit	Bitti	1	0-1	1
oNibble	Puolitavu	4	0-15	1
oByte	Tavu	8	0-255	2
oWord	Sana	16	0-65535	3

Jokainen määritetty muuttuja vie tietyn määrän OOPic:n oliomuistia ja tästä syystä ennen muuttujan määrittelyä tulee selvittää suurin arvo, minkä muuttuja voi saada ja valita pienin muuttujatyyppe, johon haluttu arvo mahtuu.

Kaikissa näissä muuttujissa on lippu-tyyppinen NonZero-ominaisuus, joka on päällä, kun muuttujan arvo ei ole 0.

oByte- ja oWord-muuttujissa on lippu-tyyppinen MSB-ominaisuus, joka on päällä, kun muuttujan arvon eniten merkitsevä bitti (*Most Significant Bit*) on 1.

oBit-muuttujalla on muista poiketen lippu-tyyppinen oletusominaisuus.

Lippu-tyyppiset ominaisuudet voidaan linkittää virtuaalipiirien lippu-osoittimiin. (Katso kappale 9, "Virtuaalipiirit" lisätietoja lippu-tyyppisten ominaisuuksien käytöstä virtuaalipiirien yhteydessä)

### **7.3 oBuffer-muuttuja**

oBuffer-muuttujaa käytetään peräkkäisten tavujen tallettamiseen. Sen koko voidaan määrittää 1-32 tavuksi.

oBuffer-muuttujan Value-ominaisuus on yhden tavun mittainen ja osoittaa yhtä elementtiä peräkkäisten tavujen joukosta. Elementin osoittama paikka asetetaan Location-ominaisuuden avulla.

### **7.4 oEEProm-muuttuja**

oEEProm-muuttujaa käytetään tallettamaan ja hakemaan tietoa haihtumattomasta muistista EEPROM muistipiiristä, joka on kytketty OOPicin E0 (tai II+-mallissa E1) paikkaan.

Oliota käytettäessä kannattaa huomioida, että OOPicin ohjelma sijaitsee E0-muistissa ja voit kirjoittaa käynnissä olevan sovelluksen päälle ja sekoittaa sen toiminnan.

### **7.5 oRam-muuttuja**

oRam-muuttujaa käytetään 256 OOPicin prosessorissa olevan muistipaikan käsittelyyn.

Oliota käytettäessä kannattaa huomioida, että OOPicin käyttöjärjestelmä sijaitsee RAM-muistissa ja voit sekoittaa OOPicin toiminnan.

### **7.6 Vakiot**

Vakio on arvo, jolla on nimi. Kuten muuttujaa, vakiota voidaan käyttää ohjelmakoodissa sen arvon sijaan. Toisin kuin muuttujalla, vakion arvo voidaan määrittää vain kerran ohjelmakoodissa ja sen arvoa ei pysty muuttamaan ohjelman suorituksen aikana.

## 8 Liittäminen laitteisiin

### 8.1 OOPicin laitteistoliityntä

OOPic on suunniteltu käytettäväksi sulautetuissa sovelluksissa. Näissä sovelluksissa yksittäinen prosessori ohjaa jonkinlaista laitteistoa. Tämän takia OOPicissä on 31 I/O(*Input/Output*) -linjaa, joita voidaan käyttää usealla eri tavalla. Kukin I/O-linja voi ottaa vastaan tai antaa ulos 25mA virran. I/O-linjoissa 8 - 15 on sisäiset ylösvetovastukset, jotka voidaan aktivoida OOPic-olion Pullup-ominaisuudella. I/O-linjoissa 16 - 31 on Schmidt Trigger -tyyppiset tulot, kun ne on määritetty input-tilaan. Näitä I/O-linjoja hallitaan joukolla olioita, joita kutsutaan laitteisto-olioiksi.

Seuraavassa taulukossa on kuvattuna OOPicin I/O-linjojen sähköiset ominaisuudet.

Lähdön jännite, kun I/O-linja on asetettu lähdeksi	0. 0 Volttia
Maksimivirta, mitä lähdestä tulee ulos, kun I/O-linja on asetettu lähdeksi	0. 25 mA
Lähdön jännite, kun I/O-lähtölinja on asetettu 1:ksi.	+5 Volttia
Maksimivirta, mitä lähdestä tulee ulos, kun I/O-lähtölinja on asetettu 1:ksi.	25 mA
Maksimivirta, mitä kaikista lähdeistä saadaan yhteensä	200 mA

OOPic II+:n jänniteregulaattori on TO-92 -koteloinen 7805, +5 voltin ja 100 mA:n regulaattori. Mikäli tarvitaan enemmän virtaa, regulaattori voidaan vaihtaa tehokkaampaan tai reguloitu +5 voltin jännitelähde voidaan kytkeä OOPicin I/O-liittimen +5 voltin liitäntään.

Laitteisto-oliot ovat olioita, jotka yhdistävät fyysiset laitteistot OOPiciin. Ne ovat ainoa tapa, minkä kautta OOPicin sovellusohjelmilla on mahdollisuus ohjata laitteita ja vastata laitteistossa tapahtuviin tapahtumiin.

### 8.2 Laitteisto-olioiden käyttö

Kuten kaikilla olioilla, laitteisto-oliot täytyy luoda ennen käyttöä. Basic-kielessä tämä tapahtuu Dim-lausekkeella. Dim-lauseke saa parametreikseen yksilöllisen nimen oliolle ja luokkamäärittelyn, joka kertoo millainen olio luodaan. Lisätietoa Dim-lausekkeesta löytyy kappaleesta 3.

Luettelo kaikista laitteisto-olioista löytyy OOPicin olio-luettelosta osoitteesta <http://www.oopic.com/objlist.htm>

## 9 Virtuaalipiirit

### 9.1 Mikä on virtuaalipiiri?

Virtuaalipiiri (*Virtual Circuit*) on OOPic:n toiminto, joka näyttää toiminnallisesti diskreetiltä elektroniikkapiiriltä, mutta on itse asiassa OOPic käyttäjärjestelmä, joka emuloi piirin toimintaa.

Virtuaalipiiri voi sisältää minkä tahansa tyyppisiä olioita kunhan ainakin yksi käsittelyolio (*Processing Object*) on mukana. Jokaista yksittäistä virtuaalipiirin osaa voidaan käsitellä ja arvioida ohjelmallisesti, jolloin koko piiri on hallittavissa.

### 9.2 Kuinka virtuaalipiirit tehdään?

Virtuaalipiirit tehdään linkittämällä yhten joukko olioita samalla tavalla kuin joukko elektroniikkapiirejä liitettäisiin fyysisesti yhteen sähköpiirin kokoamiseksi. Sovellusohjelma voi koota, purkaa tai uudelleenkonfiguroida virtuaalipiirin rakneteen missä tahansa ohjelman suorituksen vaiheessa.

Olioita, joita käytetään linkittämään yhteen virtuaalipiirin osia, saavat tuloarvonsa ja tallettavat lähtöarvonsa toisiin olioihin. Näitä olioita kutsutaan käsittelyolioiksi (*Processing Objects*). Nämä oliot sisältävät erilaisia matemaattisia, loogisia ja muita tiedonkäsittelyfunktioita. Kun ne linkitetään toisiin olioihin, niiden sisältämät operaatiot suoritetaan linkitetyille olioille. Linkki, joka määrittää mitä olioita käytetään, tehdään olion ominaisuuden nimeltä *Pointer* (osoitin) kautta.

### 9.3 Olioiden linkittäminen

Linkitettäessä ohjelmallisesti virtuaalipiiriin olioita toisiinsa käytetään *Link*-metodia yhdessä yhdistettävien olioiden ja niiden ominaisuuksien kanssa. Näin muodostetaan linkki kahden olion välille. Kun linkki on luotu, linkitty olio saa arvon linkitetystä ominaisuudesta.

Muiden metodien käynnistäen oliossa jonkin operaation suorituksen *Link*-metodia käytetään yhdistämään *Pointer*-ominaisuus toisen olion johonkin ominaisuuteen. *Link*-metodin syntaksi on seuraava:

```
perusolio.osoitinOminaisuus.Link(linkkiolio ominaisuusJohonLinkitetaan)
```

*Link*-metodi on käytettävissä ainoastaan ominaisuuksissa, jotka on suunniteltu linkitettäväksi.

### 9.4 Osoittimen ominaisuudet

Osoitin-ominaisuus on olion ominaisuus, joka kertoo kyseiselle oliolle, mistä tieto toisessa oliossa löytyy. Se ei itse talleta tietoa vaan joka kerta, kun tietoa tarvitaan, osoitin-ominaisuutta käytetään hyväksi määrittäessä paikkaa, mistä tieto haetaan.

Olion ominaisuus, johon osoitin-ominaisuus on linkitetty, voidaan vaihtaa missä tahansa vaiheessa ohjelman suoritusta.

On olemassa kahden eri tyyppin osoitin-ominaisuuksia:

- Osoitin kohdeolion oletus-ominaisuuteen, kutsutaan nimellä olio-osoitin.
- Osoitin kohdeolion johonkin lippu-tyyppiseen ominaisuuteen, kutsutaan nimellä lippu-osoitin.

Kun osoittimen ominaisuus linkitetään olion ominaisuuteen, täytyy osoittimen ominaisuuden tyyppin ja olion ominaisuuden tyyppin olla samoja.

## 9.5 Olio-osoittimet

Olio-osoitinta käytetään linkittämään käsittelyolio (*Processing object*) toisen olion oletusominaisuuteen. Linkitettäessä argumentti, jota sulkujen sisään odotetaan Link-metodin jälkeen, on kohdeolion nimi, piste ja kohdeolion oletusominaisuus tai pelkästään kohdeolion nimi.

Oletusominaisuus on se olion ominaisuus, jota käytetään, jos mitään ominaisuutta ei määritetä viitattaessa olioon. Tieto kunkin olion oletusominaisuudesta löytyy olion teknisistä tiedoista, missä kunkin ominaisuuden tietotyypit määritetään.

Seuraavassa esimerkissä oMath-olion olio-osoittimet linkitetään kahden muun olion oletusominaisuuksiin.

```
Dim a As New oMath      ' Luo oMath käsittelyolio
Dim b As New oByte      ' Luo 1 tavun muuttujaolio
Dim c As New oDio8      ' Luo 8-bittinen I/O portti

Sub Main()
    a.Input1.Link(b.Value) ' Linkitse 1. oMath tulo
                          ' oByte olioon
    a.Input2.Link(c.Value) ' Linkitse 2. oMath tulo
                          ' I/O-porttiin

End Sub
```

## 9.6 Lippuosoittimet

Lippuosoittimia käytetään linkittämään käsittelyolio (*Processing Object*) toisen olion lippu-tyyppiseen ominaisuuteen. Linkitettäessä argumentti, jota sulkujen sisään odotetaan Link-metodin jälkeen, on kohdeolion nimi, piste ja haluttu kohdeolion lippu-tyyppinen ominaisuus.

Lippu-tyyppinen ominaisuus on ominaisuus, joka muodostuu yhdestä bitistä ja on konfiguroitu siten, että siiden voidaan linkittää. Tieto kunkin olion

lippu-tyyppisistä ominaisuuksista löytyy olion teknisistä tiedoista, missä kunkin ominaisuuden tietotyypit määritetään.

Seuraavassa esimerkissä oGate-olion lippuosoittimet linkitetään kahden muun olion lipputyypisiin ominaisuuksiin.

```
Dim a As New oGate      ' Luo oGate-käsittelyolio
Dim b As New oByte      ' Luo 1 tavun muuttujaolio
Dim c As New oDio1      ' Luo 1 bitin I/O-portti

Sub Main()
    a.Input1.Link(b.NonZero)    ' Linkitse 1. oGaten tulo
                                ' 1. I/O-porttiin
    a.Output.Link(c.Value)      ' Linkitse oGaten lähtö 2.
                                ' I/O-porttiin
End Sub
```

## 10 OOPic järjestelmäolio

### 10.1 Mikä on OOPic-olio?

OOPic-olio on järjestelmän sisäinen olio, joka hallitsee OOPicin käyttöjärjestelmää. Sen ominaisuudet joko hallitsevat käyttöjärjestelmää tai raportoivat käyttöjärjestelmän tilasta. OOPic-olio on ainoa järjestelmän sisäinen olio.

Koska OOPic-olio on järjestelmän sisäinen olio, se on aina mukana sovelluksessa eikä ohjelmoijan tarvitse huolehtia sen muistin varaamisesta.

### 10.2 OOPicin käyttöjärjestelmän hallinta

Neljä ominaisuutta liittyy OOPicin käyttöjärjestelmän hallintaan:

- Node
- Operate
- Pause
- Reset

Node (*Solmu*)-ominaisuus on arvo, jota käytetään laitteiden tunnistamiseen, kun kaksi tai useampia laitteita on kytketty I2C-verkkoon.

Operate-ominaisuuden arvo määrittää OOPic-piirin virran tilan.

Pause-ominaisuus keskeyttää ohjelman suorituksen.

Reset-ominaisuus resetoi asetettaessa OOPicin.

### 10.3 OOPicin käyttöjärjestelmän tilan lukeminen

Kolme ominaisuutta raportoi järjestelmän tilasta.

- Hz1
- Hz60
- StartStat

Hz1-ominaisuus vaihtuu kerran sekunnissa eli 1 Hz taajuudella.

Hz60-ominaisuus vaihtuu kerran 1/60 sekunnissa eli 60 Hz taajuudella.

StartStat-ominaisuus kertoo viimeisen resetoinnin syyn.

### 10.4 Muiden OOPicin toimintojen hallinta

Kaksi ominaisuutta määrittävät OOPicin laitteiston toimintaa.

- ExtVRef
- PullUp

ExtVRef-ominaisuus määrittää, mitä OOPicin AD-muunnin käyttää referensijännitteen lähteenään.

PullUp-ominaisuus määrittää käytetäänkö sisäisiä ylösvetovastuksia I/O-linjoille 8 - 15.

# 11 OOPicin matematiikka

## 11.1 OOPicin matematiikasta

Mikä tahansa matemaattinen yhtälö, joka sovellusohjelmassa tulee ratkaista, tulee kirjoittaa lausekkeeksi ohjelmakoodiin.

## 11.2 Lausekkeet

Lauseke (*Expression*) on kirjoitettu matemaattinen kaava, joka sisältää arvoja, operaattoreita ja funktioita sekä niiden suoritusjärjestyksen, että kaava voidaan ratkaista. Tuloksena on arvo, joka saadaan, kun kaikki operaatiot ja funktiot on suoritettu

Lauseke voi olla yksittäinen arvo tai muuttuja tai se voi muodostua useista arvoista ja muuttujista, joita operaattorit ja funktiot erottavat.

Lausekkeita voidaan käyttää seuraavissa tilanteissa:

- Sijoituslausekkeessa, jossa muuttujalle annetaan arvo
- Vuonohjauslausekkeessa, jossa ohjelman seuraavaksi suoritettava toiminto riippuu lausekkeen tuloksesta.

Jos lauseke muodostuu useammasta kuin yhdestä arvosta tai muuttujasta, sen arvo määritetään vasemmalta oikealle laskien ottaen huomioon operaattoreiden ja funktioiden laskujärjestys.

## 11.3 Operaattorit

Operaattorit määrittävät mitä matemaattisia operaatioita arvojen välillä suoritetaan. Operaattorit suoritetaan aina siten, että vasemman puolen arvoksi tulee operaattorin määrittämällä tavalla muokattu oikean puolen arvo.

Seuraavassa lista operaattoreista.

- Aritmeettiset: +, -, \*, /, mod
- Loogiset: And, Or, Xor, Not
- Vertailuoperaattori: =, <>, <, >, <=, >=

Seuraavassa esimerkissä muuttujan "A" arvoksi sijoitetaan muuttujan "B" arvo plus muuttujan "C" arvo miinus muuttujan "D" arvo

$A = B + C - D$

## 11.4 Operaatioiden suoritusjärjestys

Operaatioiden suoritusjärjestys määrittää sen järjestyksen, missä matemaattiset operaatiot suoritetaan. OOPicissä suoritusjärjestys noudattaa aritmetiikan suoritusjärjestystä. Näiden sääntöjen mukaan lauseke käydään läpi vasemmalta oikealle ja operaatiota ei suoriteta kunnes kaikki korkeamman tason operaatiot on suoritettu.

Seuraavassa on OOPicin käyttämä operaatioiden suoritusjärjestys:

1. Suluissa olevat operaatiot
2. Etumerkki (-)
3. Kertolasku (\*), jakolasku (/) ja jakojäännös (Mod)
4. Yhteenlasku (+) ja vähennyslasku (-)
5. Vertailuoperaatiot (=, <>, <, >, <=, >=)
6. Looginen AND (And)
7. Looginen OR (Or)
8. Looginen XOR (XOr)

Suluissa olevilla lausekkeilla on korkein prioriteetti, joten niiden käyttö selventää suoritusjärjestystä ja tekee ohjelmasta luettavampaa.

Seuraavassa esimerkissä muuttuja "A" saa arvon, joka on muuttujan "C" arvo jaettuna muuttujan "D" arvolla plus muuttujan "B" arvo

$$A = B + C / D$$

Seuraavassa esimerkissä muuttuja "A" saa arvon, joka on muuttujien "B" ja "C" arvojen summa jaettuna muuttujan "D" arvolla

$$A = (B + C) / D$$

### 11.5 Aritmeettiset operaattorit

Aritmeettiset operaattorit suorittavat peruslaskusuorituksia kahden lausekkeen välillä.

Syntaksi	Tulos
{lauseke1} + {lauseke2}	Kahden lausekkeen summa.
{lauseke1} - {lauseke2}	Kahden lausekkeen erotus.
{lauseke1}* {lauseke2}	Kahden lausekkeen tulo
{lauseke1} / {lauseke2}	Kahden lausekkeen jakolasku.
{lauseke1} Mod {lauseke2}	Kahden lausekkeen jakolaskun jakojäännös.

### 11.6 Loogiset Operaattorit

Loogiset operaattorit tekevät biteittäisiä loogisia operaatioita kahden lausekkeen välillä.

Syntaksi	Tulos
{lauseke1} And {lauseke2}	Tulos on kahden lausekkeen looginen AND-operaatio biteittäin.
{lauseke1} Or {lauseke2}	Tulos on kahden lausekkeen looginen OR-operaatio biteittäin.

$\{ \text{lauseke1} \} \text{ Xor } \{ \text{lauseke2} \}$  Tulos on kahden lausekkeen looginen XOR-  
operaatio biteittäin.

Not  $\{ \text{lauseke1} \}$  Tuloksena  $\{ \text{lauseke1} \}$ :n bitit invertoidaan

## 11.7 Vertailuoperaattorit

Vertailuoperaattorit suorittavat kahden lausekkeen arvojen vertailua.

Syntaksi	Tulos
$\{ \text{lauseke1} \} = \{ \text{lauseke2} \}$	1 jos kaksi lauseketta ovat yhtäsuuria, 0 jos ei.
$\{ \text{lauseke1} \} <> \{ \text{lauseke2} \}$	1 jos kaksi lauseketta ovat erisuuria, 0 jos ei.
$\{ \text{lauseke1} \} < \{ \text{lauseke2} \}$	1 jos $\{ \text{lauseke1} \}$ on pienempi kuin $\{ \text{lauseke2} \}$ , 0 jos ei.
$\{ \text{lauseke1} \} > \{ \text{lauseke2} \}$	1 jos $\{ \text{lauseke1} \}$ on suurempi kuin $\{ \text{lauseke2} \}$ , 0 jos ei.
$\{ \text{lauseke1} \} \leq \{ \text{lauseke2} \}$	1 jos $\{ \text{lauseke1} \}$ on pienempi tai yhtäsuuri kuin $\{ \text{lauseke2} \}$ , 0 jos ei.
$\{ \text{lauseke1} \} \geq \{ \text{lauseke2} \}$	1 jos $\{ \text{lauseke1} \}$ on suurempi tai yhtäsuuri kuin $\{ \text{lauseke2} \}$ , 0 jos ei.

## 11.8 Funktiot

Funktioita käytetään muuntamaan arvo muodosta toiseen. Niiden suoritus tapahtuu välittämällä suluissa oleva arvo funktion nimen määrittämälle muunnosprosessille.

Tällä hetkellä käytettävissä on yksi funktio.

Syntaksi	Tulos
$\text{Sin}(\{ \text{lauseke} \})$	Binäärisesti muotoiltu trigonometrinen sini $\{ \text{lauseke} \}$ :sta

## 12 Dynaaminen tiedonsiirto (DDE)

### 12.1 Mitä on Dynaaminen tiedonsiirto?

Dynaaminen tiedonsiirto (*Dynamic Data Exchange*) on toiminto, jonka avulla kaksi tai useampia OOPicejä voi vaihtaa tietoja keskenään. Tiedonsiirto tapahtuu virtuaalipiirillä, joka käyttää oDDELink-oliota tiedonsiirtoon I2C-verkon kautta.

Kun muita käsittelyolioita käytetään muokkaamaan ja siirtämään tietoa yhden ja saman OOPicin sisällä, oDDELink-olio on suunniteltu hyödyntämään I2C-verkkoa ja vaihtamaan tietoa muiden OOPicien kanssa.

### 12.2 oDDELink-olio

oDDELink-olio on käsittelyolio, joka siirtää tietoa kahden I2C-verkkoon kytketyn OOPicin välillä. Olio voidaan konfiguroida joko lähettämään tai vastaanottamaan tietoa.

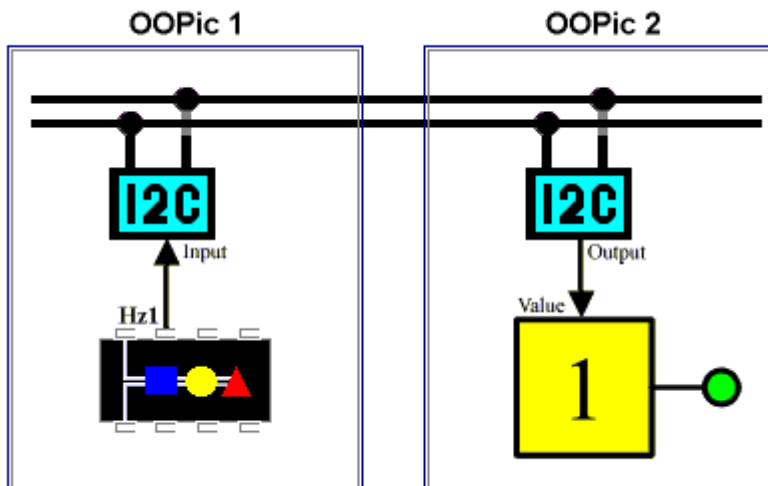
Kun oDDELink-olio on konfiguroitu, se voi aloittaa tiedonsiirron kun sen Sync-ominaisuus asetaan 1:ksi (cvTrue). Tällöin tietoa joko lähetetään tai vastaanotetaan toisesta verkossa olevasta oDDELink-oliosta riippuen Direction-ominaisuuden arvosta.

### 12.3 DDE-virtuaalipiiri

oDDELink-olion toimintojen ymmärtäminen edellyttää virtuaalipiirin toiminnan ymmärtämistä. Jos et vielä tunne virtuaalipiirejä, lue kappale 3 - Ensimmäinen virtuaalipiiri ja kappale 9 - Virtuaalipiirit.

Jotta virtuaalipiiri voisi siirtää tietoa I2C-verkon kautta, oDDELink-olio täytyy linkittää OOPicin muihin olioihin. Jokaisella oDDELink-olion ilmentymällä on kaksi osoitinta, joita käytetään linkittämään oDDELink-olio olioihin, joissa siirrettävä tieto sijaitsee.

oDDELink-olio saa lähetettävän tiedon oliolta, joka on linkitetty Input-ominaisuuteen ja Output-ominaisuuteen linkitetään olio, johon vastaanotettu tieto talletetaan.



oDDELink-olion Sync-ominaisuus, mikä käynnistää tiedonsiirron, on lippu-tyyppinen ominaisuus ja sitä voidaan ohjata virtuaalipiirillä linkittämällä se johonkin virtuaalipiiriin lippu-osoittimeen.

## 12.4 Verkon solmut

Jokainen I2C-verkkoon kytketty OOPic täytyy erotella toisistaan yksilöllisellä solmunumerolla. OOPic-olion Node-ominaisuuteen talletetaan solmun numero. Tätä tunnusta I2C-verkko käyttää tiedon siirtämiseen oikealle laitteelle, kun DDELink-tiedonsiirtoa tehdään.

OOPic.Node-ominaisuus täytyy asettaa isommaksi kuin 0 OOPicin I2C-verkkotoimintojen käynnistämiseksi. Kun solmun numero on asetettu, I2C-verkko käynnistyy ja OOPic alkaa kuuntelemaan tulevia paketteja. Jos OOPic.Node asetetaan takaisin 0:ksi (cvOff), I2C-verkkotoiminnot pysäytetään ja kaikki tulevat tietopaketit jätetään huomioimatta.

I2C-verkko on käynnistettävä sekä isäntäsovelluksessa (*Master*) että orjasovelluksessa (*Slave*) ennen kuin sovellukset yrittävät siirtää tietoa.

oDDELink-oliossa on myös Node-ominaisuus. Tämä määrittää minkä OOPicin kanssa oDDELink-olio tulee keskustelemaan.

## 12.5 DDE-keskustelu

Kun kaksi OOPiciä vaihtavat tietoa, ne tekevät sen DDE-keskustelun avulla. Sovellusta, joka aloittaa keskustelun, kutsutaan isäntäsovellukseksi (*Master-Application*) tai pelkästään isännäksi (*Master*); ja sovellus, joka vastaa isännälle, on orjasovellus (*Slave-Application*), tai pelkästään orja (*Slave*). Sama sovellus voidaan määritellä toimimaan useassa keskustelussa toimien isäntänä toisissa ja orjana toisissa keskusteluissa.

Yksittäinen oDDELink-olio voi toimia sekä isäntänä että orjana DDELink-keskustelussa. Lisäksi se voi toimia isäntänä DDELink-keskustelussa yhden

OOPicin kanssa ja orjana kokonaan erillisessä DDELink-keskustelussa toisen OOPicin kanssa.

DDELink-keskustelu käynnistyy, kun oDDELink-olioon, jota aiotaan käyttää isäntänä, tehdään tarvittavat asetukset: orjan Node- ja Location-ominaisuudet talletetaan oDDELink-olioon ja Operate-ominaisuus asetetaan 1:ksi. Kun tämän jälkeen isännän Sync-ominaisuus asetetaan 1:ksi, käynnistyy DDELink-keskustelu, mikä synkronoi tiedon isännän ja orjan välillä.

## 12.6 DDE-isäntä

oDDELink-isäntäolio hallitsee aina DDE-keskustelua. Isäntä käynnistää aina keskustelun riippumatta siitä onko se lähettämässä vai vastaanottamassa tietoa.

Jotta isäntä voisi käynnistää DDE-keskustelun siihen täytyy määrittää mihin OOPiciin yritetään ottaa yhteyttä ja orjan oDDELink-olion sijainti.

Isännän oDDELink-olion Node-ominaisuus täytyy laittaa samaksi numeroksi kuin orjan OOPic.Node-ominaisuus on asetettu.

Orjan oDDELink-olion sijainti täytyy asettaa isännän oDDELink-olion Location-ominaisuuteen. Se saadaan sijoittamalla siihen orjan oDDELink-olion Address-ominaisuuden arvo. Huom! Address-ominaisuuden arvo määräytyy, kun ohjelma käännetään ja sovellus ei voi sitä muuttaa. Nopein tapa Address-ominaisuuden arvon saamiseksi on kääntää orjan koodi ja valita View-alasvetovalikosta *Opp Codes*. Siellä oDDELink-olion vasemmalla puolella näkyy sen osoite, kuvassa ympyröitynä 41.

```

C:\Program Files\OOPic\leddde.osc
Dim Slave as New oDDELink
Dim Led as New oDio1

Sub Main()
  OOPic.Node=2
  Led.IOLine=31
  Led.Direction=cvOutput
  Slave.Output.Link(Led.Value)
  Slave.Operate=cvTrue
End Sub

-----:<@Dim-Begin>:
-----:<@Dim#1-Begin>:
0020:041 '->Slave
0021:001 'ArraySize
0022:008 'ObjSize
0023:007 'ObjID:oDdelink
0024:135 'Dim
-----:<@Dim#1-Exit>:
-----:<@Dim#2-Begin>:
0025:049 '->Led
0026:001 'ArraySize
  
```

Jotta isäntä voisi lähettää tietoa, sen Input-ominaisuus täytyy linkittää olioon, joka sisältää lähetettävän tiedon. Ja jotta isäntä voisi vastaanottaa tietoa, sen Output-ominaisuus täytyy linkittää olioon, johon tullut tieto talletetaan.

### 12.6.1 Kuinka isännän oDDELink-olio toimii

Kun oDDELink-olion Sync-ominaisuus asetetaan 1 (cvTrue), se ensi tarkistaa onko sen Operate-ominaisuus myös 1 (cvTrue). Jos näin ei ole, oDDELink-olio säilyy lepotilassa. Jos Operate-ominaisuus on 1, niin Node- ja Location-ominaisuudet tarkistetaan. Jos jompikumpi ominaisuus on 0, niin silloin

operaatio perutaan. Jos niiden arvo taas on suurempi kuin 0, DDELink-keskustelu käynnistetään.

Kun DDELink-keskustelu käynnistetään, isäntä tarkistaa Direction-ominaisuuden asetuksen. Jos se on 0 (cvSend), niin isäntä lähettää tietoa orjalle, ja jos se on 1 (cvReceive), niin isäntä vastaanottaa tietoa orjalta.

Kun isäntä lähettää tietoa orjalle, se hakee tiedon siltä oliolta, johon sen Input-ominaisuus on linkitetty. Isäntä lähettää tiedon orjalle, joka tallettaa sen olioon, johon sen Output-ominaisuus on linkitetty.

Kun isäntä vastaanottaa tietoa orjalta, se lähettää ensi pyynnön orjalle hakea tieto siltä oliolta, johon orjan Input-ominaisuus on linkitetty. Isäntä tallettaa orjalta saamansa tiedon olioon, johon sen Output-ominaisuus on linkitetty.

Seuraavassa on esimerkki OOPic sovelluksesta, joka käyttää Master oDDELink-oliota yksinkertaisessa virtuaalipiirissä.

`Dim Master As New oDDELink`

```
Sub Main()  
    OOPic.Node = 1  
    Master.Input.Link(OOPic.Hz1)  
    Master.Node = 2  
    Master.Location = 43  
    Master.Direction = cvSend  
    Master.Operate = cvTrue  
    Do  
        If Master.Tranmitting = cvFalse then  
            Master.Sync = 1  
        End If  
    Loop  
End Sub
```

## 12.7 DDE-orja

Orjan oDDELink-olio jatkuvasti tarkkailee I2C-verkkoa DDELink-keskustelun varalta. Jos se ei havaitse isännän DDELink-keskustelupyynnö, se pysyttelee lepotilassa.

Jotta orja voisi osallistua DDELink-keskusteluun, sen Input- ja Output-ominaisuuksien tulee olla asetettuina. Orjan Operate-ominaisuuden ei

tarvitse olla asetettuna 1:ksi (cvTrue), että se pystyisi vastaamaan isännän ohjeisiin.

### 12.7.1 Kuinka orjan oDDELink toimii

Kun orjan oDDELink-olio osallistuu DDELink-keskusteluun isännän kanssa, se joko lähettää isännälle tietoa tai vastaanottaa isännältä tietoa.

Jos isäntä pyytää orjalta tietoa, niin orja vastaa hakemalla tiedon oliolta, johon sen Input-ominaisuus osoittaa ja lähettämällä tiedon sitten isännälle.

Jos isäntä lähettää tietoa orjalle, orja odottaa kunnes tietopaketti on saapunut. Kun orjalla on koko paketti, orjan DDELink-olio kopioi tiedon Output-ominaisuuden osoittamaan olioon.

Seuraavassa on OOPic-sovellusesimerkki, joka käyttää orja oDDELink-olio yksinkertaisessa virtuaalipiirissä.

```
Dim SLAVE As New oDDELink
```

```
Dim LED As New oDio1
```

```
Sub Main()
```

```
    OOPic.Node = 2
```

```
    LED.IOLine = 31
```

```
    LED.Direction = cvOutput
```

```
    Slave.Output.Link(LED.Value)
```

```
    Slave.Operate = cvTrue
```

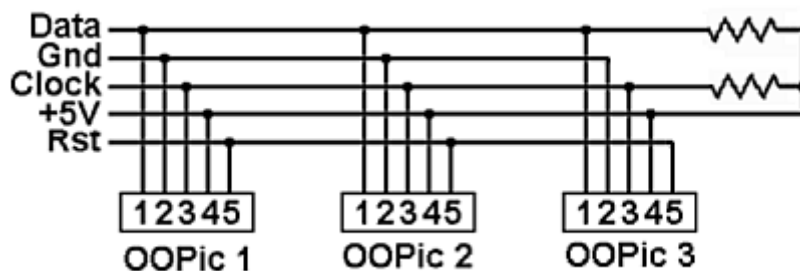
```
End Sub
```

### 12.8 I2C-verkon kytkennät

I2C-verkko on OOPiciin sisäänrakennettuna ja sen tiedonsiirtonopeus on 19,200bps. OOPicin piirilevyllä on 2 5-pinnistä verkkoliitintä, jotka koostuvat viidestä linjasta: I2C Serial Data (Data), I2C Serial Clock (Clock), maa (Gnd), +5 Volttia (+5V) ja Reset (Rst). Pinnijärjestys kahdessa 5-pinnisessä verkkoliittimessä on identtinen ja täten niitä voidaan käyttää OOPicien peräkkäinketjuttamiseen.

Kun OOPicejä kytketään yhteen I2C verkoksi, 3 linjaa; Data, Clock ja Gnd tulee kytkeä kaikissa verkon OOPiceissä yhteen, 2 linjaa; +5V ja Rst voidaan kytkeä, jos sovelluksessa tarvitaan.

Toimiakseen kunnolla sekä Data että Clock-linjat tulee kytkeä 4.7kΩ ylös vetovastuksilla +5V:n.



Jos Rst-linja on kytketty kaikissa verkossa olevissa OOPiceissä, niin painettaessa missä tahansa OOPicissä Reset-nappia kaikki verkkoon kytketyt OOPicit resetoituvat. Huom: OOPic.Reset -ominaisuuden asettaminen 1:ksi (cvTrue) jossakin verkon OOPicissä ei vedä alas Rst-linjaa ja siten resetoit kaikkia verkon OOPicejä.

## 13 Merkkijonot (*Strings*)

### 13.1 Mitä ovat merkkijonot?

Termiä "merkkijono" käytetään ohjelmistotekniikassa kuvaamaan joukkoa merkkejä, jotka on ryhmitetty yhteen ja joita käsitellään yhtenä kokonaisuutena. Vaikka OOPic on suunniteltu ohjaamaan laitteita I/O-linjojen binääristen tilojen avulla, merkkijonot ovat hyödyllisiä kun tarvitsee lähettää tekstirivejä laitteille kuten sarjaportti, tulostin tai LCD-näyttö. Merkkijonojen toteutus OOPicin kielissä on suunniteltu nimenomaan sellaiseen käyttöön.

### 13.2 Merkkijonojen käyttö

Esimerkkinä kuinka merkkijonoja käytetään voisi olla rivi, joka lähettää sanan "Terve" tulostimelle (Ptr.String = "Terve"). Se, mitä varsinaisesti riviä suoritettaessa tapahtuu, on, että 5 erillistä tiedonsiirtoa tapahtuu, yksi kullekin merkille.

Tässä esimerkissä String-ominaisuutta käytettiin merkkijonon arvon asettamiseen. String-ominaisuus on itseasiassa muokattu versio Value-ominaisuudesta, joka sallii merkkijonojen sijoituksen Value-ominaisuuden arvoksi.

Jokaisella OOPicin oliolla, jolla on Value-ominaisuus, on myös vastaava String-ominaisuus, joka mahdollistaa sen, että sovellus voi sijoittaa merkkijonon olion Value-ominaisuuden arvoksi. Esimerkiksi kun lause A.Value = 5 sijoittaa numeerisen arvon 5 olion A Value-ominaisuuden arvoksi, lause A.String = "Terve" sijoittaa peräkkäin 5 arvoa olion A Value-ominaisuuteen. Jokainen viidestä arvosta edustaa yhtä kirjainta merkkijonosta "Terve" alkaen vasemmanpuoleisimmasta merkistä. Tämä tehdään sijoittamalla ensimmäisen kirjaimen (T) ASCII-arvo olion Value-ominaisuuteen, sitten ASCII-arvo toisesta merkistä (e) ja niin eteenpäin kunnes kaikki merkkijonon kirjaimet on sijoitettu.

Kun String-ominaisuus luetaan oliosta, vastaava ASCII-arvo kuin Value-ominaisuuden arvoksi on talletettu palautetaan yhtenä merkinä. Mikäli kyseessä on 16-bittinen Value-ominaisuus, vain alemmat 8-bittiä käytetään. Sijoitusprosessin tulokset voivat riippua käytettävästä oliosta.

### 13.3 Merkkijonojen käyttö oSerial- ja oSerialPort-olioiden kanssa

Kun merkkijono sijoitetaan oSerial- tai oSerialPort -olioon, jokainen merkki, joka sijoitetaan string-ominaisuuteen lähetetään sarjamuodossa ulos I/O-linjaa pitkin.

Seuraavassa esimerkissä merkkijono "Hello World" lähetetään OOPicin sarjaliikennelinjaan kytketyllä laitteelle, kuten PC:lle.

```
Dim A As New oSerial
```

```
Sub Main()  
    A.Baud = cv9600  
    A.Operate = cvTrue  
    A.String = "Hello World"  
End Sub
```

Kun merkkijono luetaan oSerial tai oSerialPort -oliosta, palautetaan yksittäinen seuraavana saapunutta tavua vastaava merkki.

### **13.4 Merkkijonojen käyttö oDataStrobe -olion kanssa**

Kun merkkijono sijoitetaan oDataStrobe -olioon, jokainen string-ominaisuuteen sijoitettu merkki lähetetään olioon, johon se on linkitetty.

Seuraavassa esimerkissä merkkijono "Hello World" lähetetään kirjoittimeen, joka on kytketty I/O-ryhmään 3.

```
Dim Prt As New oDataStrobe  
Dim LD As New oDio8  
Dim DS As New oDio1
```

```
Sub Main()  
    LD.Iogroup = 3  
    LD.Direction = cvOutput  
    DS.Ioline = 7  
    DS.Direction = cvOutput  
    Prt.Output.Link(LD)  
    Prt.Strobe.Link(DS)  
    Prt.Operate = cvTrue  
    Prt.String = "Hello World"  
End Sub
```

Luettaessa String-ominaisuus oDataStrobe-oliosta se palauttaa aina ASCII-merkin 0.

### **13.5 Merkkijonojen käyttö oBuffer-olion kanssa**

Kun merkkijono sijoitetaan oBuffer-olioon, jokainen sijoitettu merkki täyttää yhden paikan oBuffer-olion taulukosta.

Seuraavassa esimerkissä merkkijono "Hello World" täyttää oBuffer-olion taulukon viimeistä tavua lukuunottamatta.

```
Dim A As New oBuffer(12)
```

```
Sub Main()
```

```
    A.String = "Hello World"
```

```
End Sub
```

Luettaessa oBuffer-olion String-ominaisuus palautuu koko taulukko. Yllä olleessa esimerkissä palautuu siis "Hello World" ja se ASCII-merkki, joka sattuu sijaitsemaan oBuffer-olion taulukon viimeisessä paikassa.