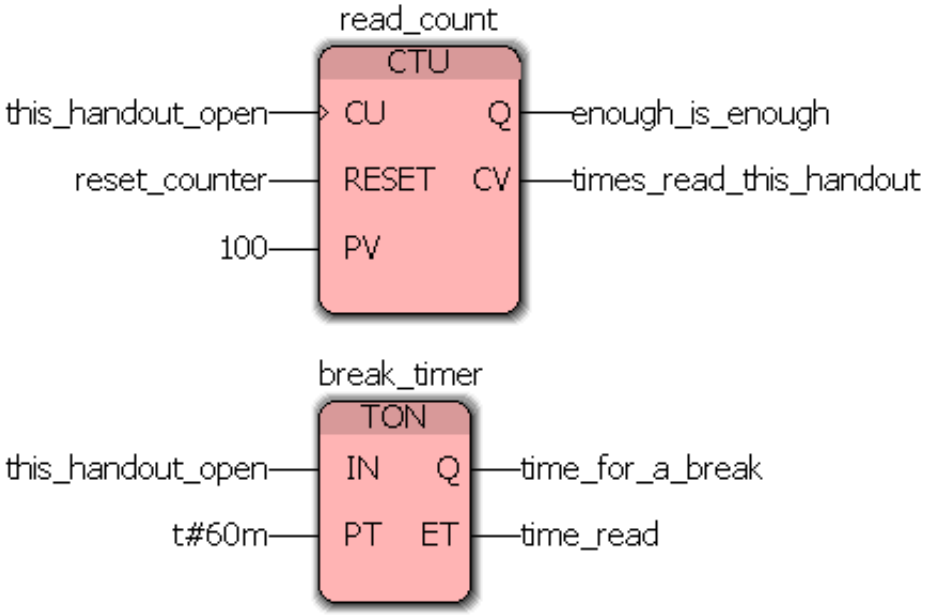
Opas toimilohko-ohjelmointiin



Automaation tietotekniikka 2011

15. elokuuta 2011

Dokumentin versio

Versio	Pvm	Muutokset	Muuttaja
0.1	8.11.2010	Ensimmäinen versio	Miika-Petteri Matikainen
0.1.1	12.11.2010	Nimetyt linkit, input duplication, korjauksia	Miika-Petteri Matikainen
0.2	20.1.2011	Omien toimilohkojen käyttö	Miika-Petteri Matikainen
0.2.1	15.8.2011	Tarkennuksia, päivityksiä	Joonas Kröger

Sisältö

1 Yleistä

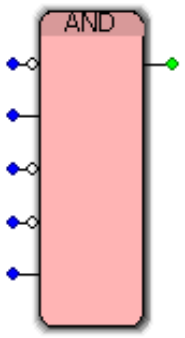
1.1 Johdanto

Tämä opas on johdatus toimilohko-ohjelmointiin FBD-kielellä (*function block diagram*, FBD). IEC 61131-3 -standardin määrittelemän FBD:n **yleisimmät toimilohkot** käydään läpi sekä annetaan esimerkkejä niiden käytöstä logiikkaohjelmassa. Kuvissa olevat ohjelmaesimerkit on tehty MULTIPROG-logiikkaohjelmointiympäristöllä.

1.2 Toimilohkot ja funktiot

Funktiot ja toimilohkot eroavat toisistaan siten, että funktioilla ei ole sisäistä tilaa. Funktioiden ulostulo on aina sama samalla syötteellä. Esimerkki funktiosta on `AND`-lohko, joka laskee sisääntulojen loogisen JA-funktion. Toimilohkoilla sen sijaan on sisäinen tila, muisti, joten toimilohkon ulostulo riippuu sekä sisääntuloista että sisäisestä tilasta. Esimerkiksi `SR`-kiikun ulostulo riippuu siitä, onko kiikku asetettu päälle aiemmin.

Lohkot[1] saattavat mahdollistaa sisääntulojen monistamisen (*duplication*), jolloin lohko voi ottaa sisäänsä oletusarvoa suuremman määrän sisääntuloja. Toisaalta sisääntulot voi joissakin lohkoissa negatoida. Negatointi vastaa tilannetta, että tulosignaali menisi ensin `NOT`-lohkon läpi ennen negatoimatonta sisääntuloa. Kuvassa 1 on `AND`-lohko, jossa on monistettu sisääntuloja ja osa sisääntuloista on negatoitu (valkoinen ympyrä sisääntulon kohdalla).



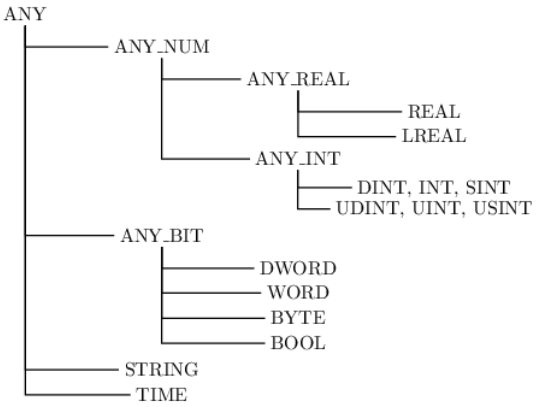
Kuva 1: AND-symboli monistetuilla muuttujilla ja osan sisääntuloista negatointi

1.3 Muuttujista

1.3.1 Tyypitys

Logiikkaohjelman muuttujat ovat tyypitettyjä ja lohkot voivat ottaa sisääntulonaan vain oikeaa tyyppiä olevia muuttujia. Esimerkiksi AND-lohko hyväksyy tyyppiä ANY_BIT olevat muuttujat (ks. kohta 3.1.1). IEC 61131-3 -standardi määrittelee tyyppihierarkian, joka on esitetty alla olevassa kuvassa:

[1] Termi *lohko* ei ole virallinen termi, mutta sitä käytetään tässä oppaassa kuvaamaan sekä funktioita että toimilohkoja



Kuva 2: Tyyppihierarkia

1.3.2 Literaalit

Muuttujien alkuarvoina ja toimilohkojen sisääntuloina voidaan käyttää *literaaleja* eli vakiomerkkijonoja kuvaamaan muuttujan arvoa. Alla olevassa taulukossa on listattuna esimerkkejä yleisimpien muuttujatyyppien mahdollisista literaaleista.

Taulukko 1: Esimerkkejä literaaleista

Tyyppi	Esimerkki
Kokonaisluku	-5 0 42 +67
Reaaliluku	-5.0 0.0 0.4299
BOOL	TRUE FALSE
STRING	” ’hello world’
TIME	t#150ms t#10s t#42m t#2h_13m

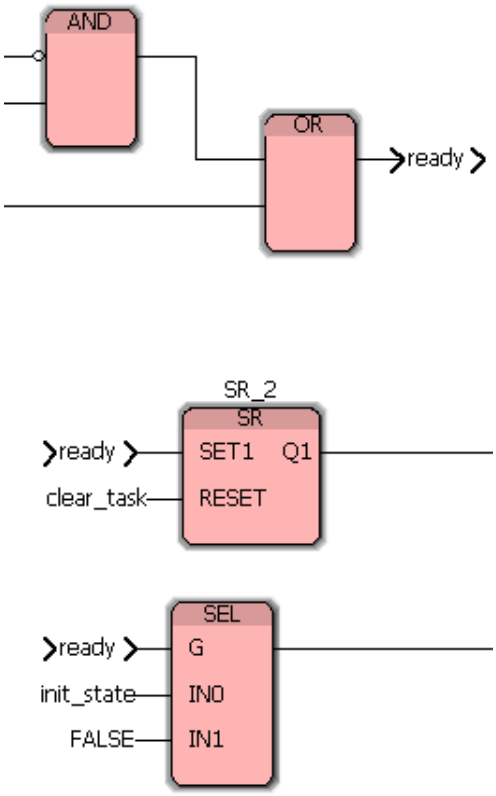
1.4 Ohjelmointityylistä

Toimilohko-ohjelma saattaa muodostua hankalasti luettavaksi, jos ohjelman rakennetta ei suunnitella kunnolla. Selkeyden vuoksi kannattaa noudattaa hyvää ohjelmointityyliä. Alla on listattu hyvän ohjelmointityylin merkkejä ja kuvassa 4 on esimerkki ohjelmasta, jossa on noudatettu hyvää ohjelmointityyliä. Kuvassa 5 on esimerkki huonosta ohjelmointityylistä, josta ei tule ottaa mallia.

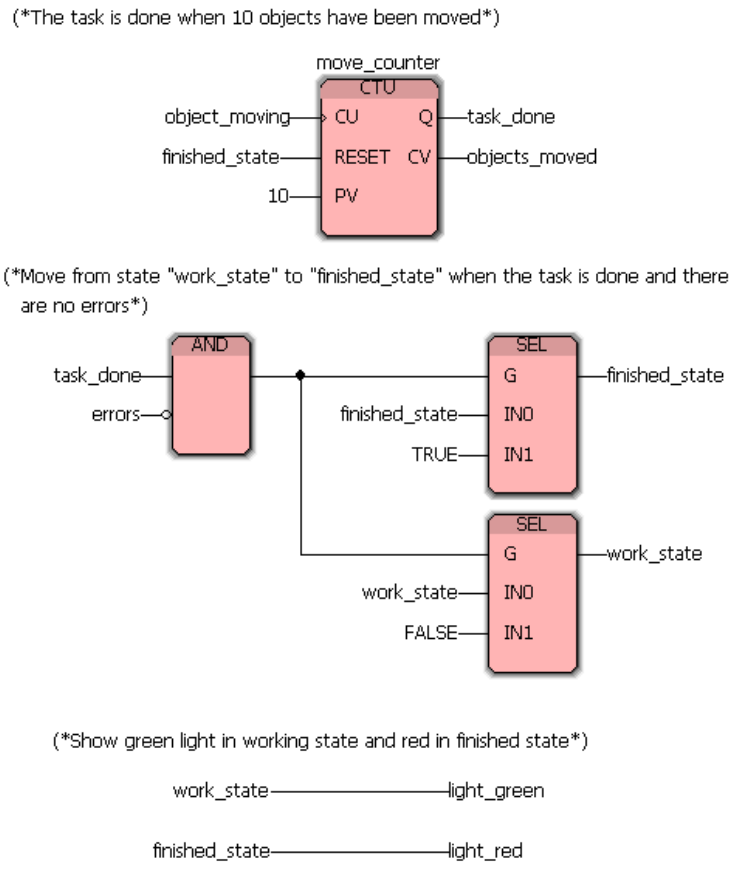
Hyvällä tyylillä muodostettu ohjelma on paitsi selkeämpi, myös helpompi päivittää ja muokata.

- Käytä kuvaavia muuttujien nimiä
- Luo väliaikaisia muuttujia tai nimettyjä linkkejä sen sijaan, että ”johdotat” saman lohkon ulostulon/sisäänmenon joka paikkaan, jossa muuttujaa tarvitaan. Väliaikaisia muuttujia voi käyttää jos lohkon ulostulo kuvaa jotain järkevää suuretta tai tilaa. Muussa tapauksessa kannattanee käyttää nimettyjä linkkkejä. Esimerkki nimetyn linkin käytöstä on kuvassa 3, jossa on luotu nimetty linkki nimeltään `ready`.

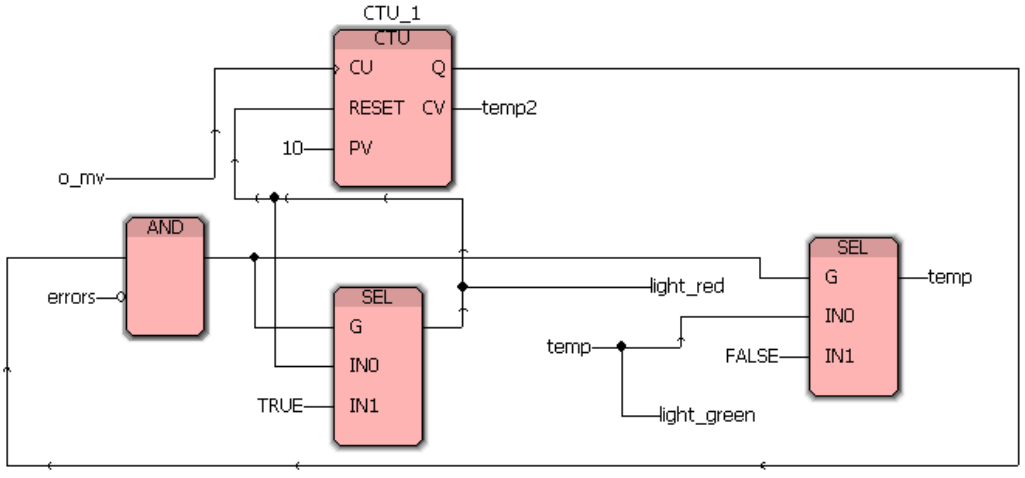
- Ohjelman ei tulisi olla liian leveä, jolloin se ei mahdu kokonaan ohjelman ruudulle. Jaa ohjelma ”riveihin”.
- Lisää kommentteja selkeyttämään ohjelmaa
- Rakenna ohjelma koostumaan selkeästi erillisistä palasista äläkä yritä ratkaista kaikkea kerralla
- Ohjelma kannattaa jakaa toimintatiloihin, joiden välillä siirrytään (esimerkkiä SEL-lohkon käytöstä tilasiirtymiin kuvassa 4). Ohjelman voi suunnitella ennen ohjelmointia esimerkiksi tilakoneena.
- Jos et enää itsekään ymmärrä miten ohjelma toimii, on se liian monimutkainen
- Kirjoita outputiin vain yhdestä paikasta



Kuva 3: Esimerkki nimetyn linkin käytöstä



Kuva 4: Esimerkki hyvästä ohjelmointityylistä



Kuva 5: Esimerkki huonosta ohjelmointityylistä

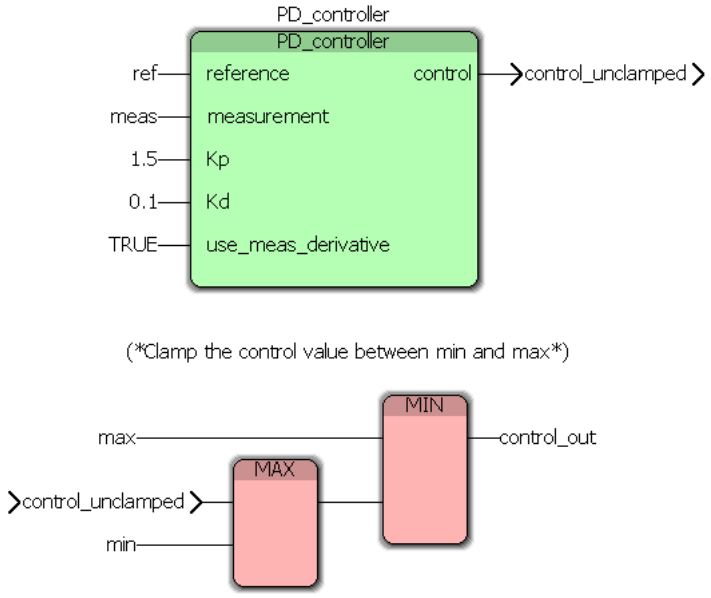
2 Omien toimilohkojen käyttäminen

Yksi keino tehdä rakenteellisesti selkeää koodia on määritellä ja käyttää omia toimilohkoja. Omia toimilohkoja voi käyttää myös abstraktointiin: piilottamaan jonkin tietyn toteutuksen ja tarjoamalla vain rajapinnan toteutukseen. Toisaalta yleiskäyttöiset omat toimilohkot voivat vähentää koodin toisteisuutta, koska samaa toteutusta ei tarvitse tehdä useampaan kertaan – riittää, että luodaan uusi toimilohko. Toiminnallisesti oma toimilohko ei eroa standardissa määritellyistä toimilohkoista; sillä on sisääntuloja, ulostuloja sekä mahdollisesti oma sisäinen tila.

Esimerkkinä oman toimilohkon käytöstä on kuvan 7 ohjelmakoodissa esiintyvä PD-säädin, joka toteuttaa seuraavat yhtälöt

$$e(n) = r(n) - y(n) \tag{1}$$
$$u_1(n) = K_p e(n) + K_d(e(n) - e(n-1)) \tag{2}$$
$$u_2(n) = K_p e(n) + K_d(y(n) - y(n-1)) \tag{3}$$

Lohkoon on määritelty sisääntulomuuttuja `use_meas_derivative`, jolla valitaan käytetäänkö derivaattatermissä erosuuretta (yhtälö 2) vai mittaussuuretta (yhtälö 3).

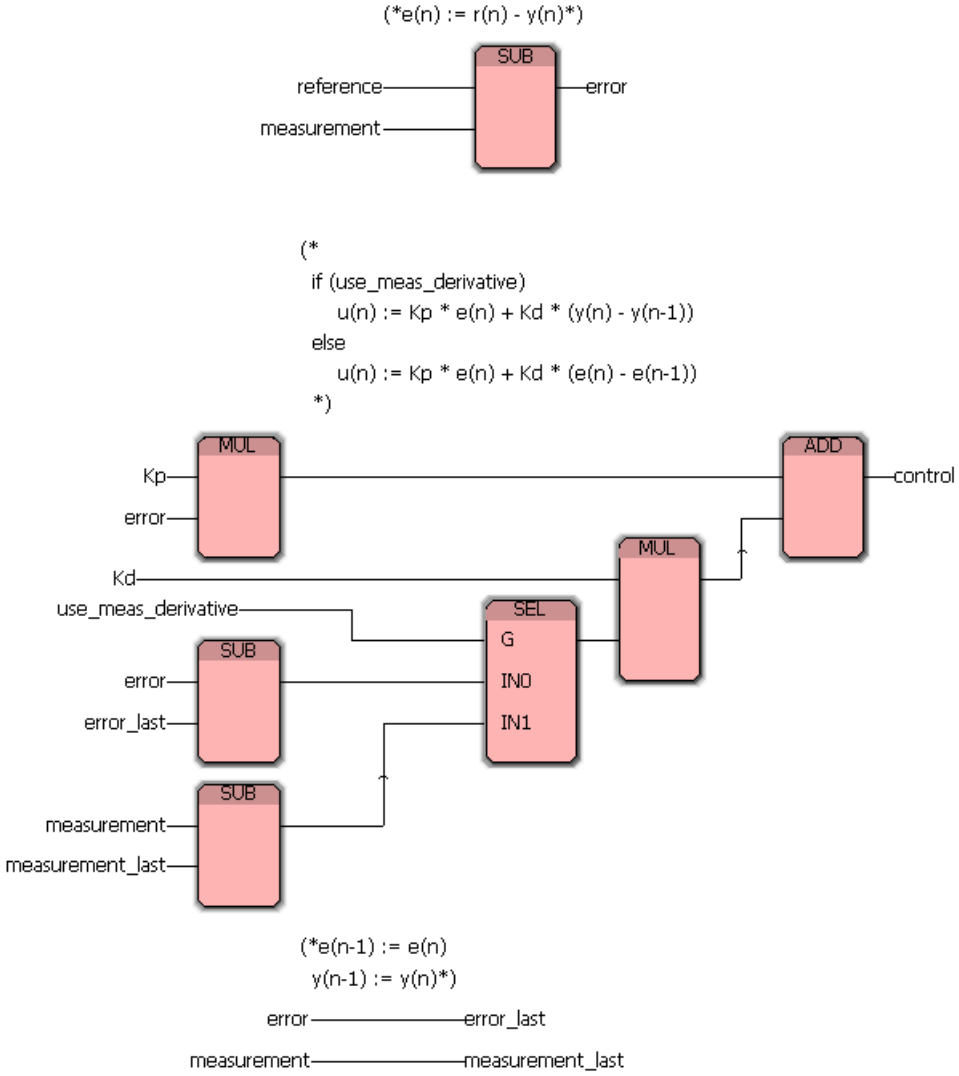


Kuva 6: Oman toimilohkon käyttö ohjelmakoodissa

Kuvassa 7 on esitetty `PD_controller`-toimilohkon sisältö. Toimilohko on ohjelmoitu FBD-kielellä[2] aivan kuten normaalin toimilohko-ohjelman tapaan. Ero-

[2]Toimilohkojen sisäisen toteutuksen voi tehdä kaikilla IEC-61131-3-standardin kielillä. Kyseinen toteutus on vain esimerkki ja todellisuudessa lohko kannattaisi tehdä ST-kielellä, jossa on tuki taulukoille (PID-säätimessä I-termiä varten täytyy tallentaa vanhat arvot). Usein PLC-järjestelmät myös tarjoavat suoraan valmista PID-lohkoa, jolloin sitä ei tarvitse itse toteuttaa

na normaaliin toimilohko-ohjelmaan on se, että toimilohkon toteutuksessa täytyy määrittää mitkä muuttujat ovat lohkon sisääntulomuuttujia (*VAR_INPUT*), mitkä ulostulomuuttujia (*VAR_OUTPUT*) ja mitkä sisäisiä muuttujia (*VAR*). Sisään- ja ulostulomuuttujat ilmenevät terminaaleja lohkossa (ks. kuva 6) ja sisäisiin muuttujiin ei pääse käsiksi lohkon ulkopuolelta.



Kuva 7: PD-säädin toteutettuna toimilohkoilla

3 Funktiot (*Functions, FU*)

3.1 Bittioperaatiot

3.1.1 AND

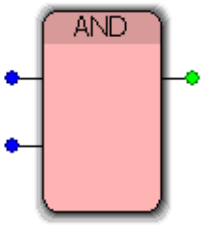
Tämä bittioperaatio toteuttaa loogisen JA-funktion sisääntuloihin yhdistetyille parametreille.

Parametri	Tietotyyppi	Kuvaus
IN1	ANY_BIT	sisääntulo
IN2	ANY_BIT	sisääntulo
OUT	ANY_BIT	ulostulo

Huom: Sisääntulo IN2 voidaan monistaa. Kaikki parametrit voidaan negatoida.
Huom: Kaikklla parametreillä on oltava sama tietotyyppi.

Totuustaulu

IN1	IN2	OUT
0	0	0
0	1	0
1	0	0
1	1	1



Kuva 8: AND-lohko

3.1.2 OR

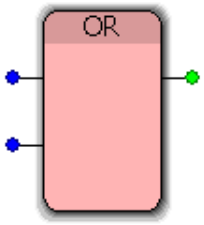
Tämä bittioperaatio toteuttaa loogisen TAI-funktion sisääntuloihin yhdistetyille parametreille.

Parametri	Tietotyyppi	Kuvaus
IN1	ANY_BIT	sisääntulo
IN2	ANY_BIT	sisääntulo
OUT	ANY_BIT	ulostulo

Huom: Sisääntulo IN2 voidaan monistaa. Kaikki parametrit voidaan negatoida.
Huom: Kaikklla parametreillä on oltava sama tietotyyppi.

Totuustaulu

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	1



Kuva 9: OR-lohko

3.1.3 XOR

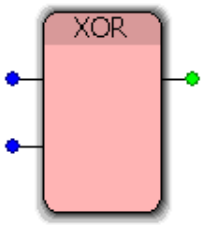
Tämä bittioperaatio toteuttaa loogisen EHDOTON TAI -funktion sisääntuloihin yhdistetyille parametreille.

Parametri	Tietotyyppi	Kuvaus
IN1	ANY_BIT	sisääntulo
IN2	ANY_BIT	sisääntulo
OUT	ANY_BIT	ulostulo

Huom: Sisääntulo IN2 voidaan monistaa. Kaikki parametrit voidaan negatoida.
Huom: Kaikklla parametreillä on oltava sama tietotyyppi.

Totuustaulu

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	0



Kuva 10: XOR-lohko

3.1.4 NOT

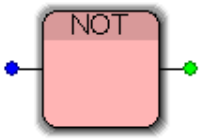
Tämä bittioperaatio negatoi sisääntulon.

Parameter	Data types	Description
IN1	ANY_BIT	sisääntulo
OUT	ANY_BIT	ulostulo

Huom: Kaikklla parametreillä on oltava sama tietotyyppi.

Totuustaulu

IN1	OUT
0	1
1	0



Kuva 11: NOT-lohko

3.2 Ehdolliset

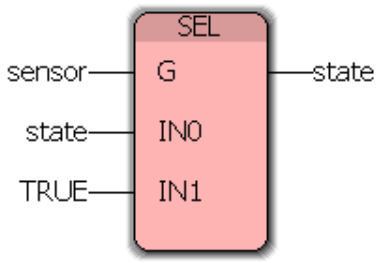
3.2.1 SEL

Tämä valintafunktio valitsee toisen kahdesta sisääntulosta riippuen sisääntulo G:n arvosta. Jos G = FALSE, ulostulon arvoksi tulee sisääntulon IN0 arvo. Jos G = TRUE, ulostulon arvoksi tulee sisääntulon IN1 arvo.

Parametri	Tietotyyppi	Kuvaus
G	BOOL	valitseva sisääntulo
IN0	ANY	sisääntulo
IN1	ANY	sisääntulo
OUT	ANY	ulostulo

Huom: Sisääntulo G voidaan negatoida.
Huom: Parametreillä IN0, IN1 ja OUT on oltava sama tietotyyppi.

Esimerkki: Jos `sensor` = TRUE, vaihdetaan parametrin `state` arvoksi TRUE, muuten `state` ei muutu.



Kuva 12: Esimerkki SEL-lohkon käytöstä

4 Toimilohkot (*Function Blocks, FB*)

4.1 Kiikut

4.1.1 SR (Set Dominant)

Tämä toimilohko toteuttaa SR-kiikun. Jos sisääntulo SET1 = TRUE, ulostulo Q1 asettuu arvoon 1. Q1 pysyy tässä arvossa vaikka SET vaihtuisi arvoon 0. Q1 resetoituu, jos RESET = TRUE. Jos molemmat sisääntulot = TRUE, ulostulo Q1 asettuu SET1:n takia arvoon TRUE.

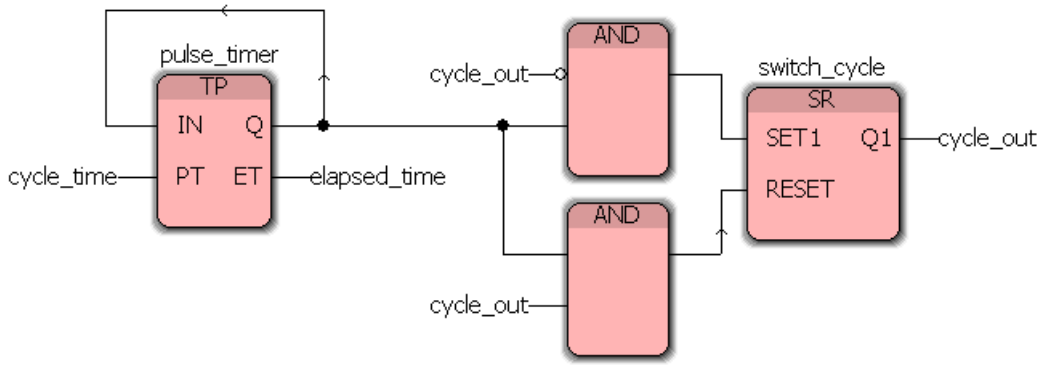
Jos toimilohkoa kutsutaan ensimmäistä kertaa, Q1 = FALSE.

Parametri	Tietotyyppi	Kuvaus
SET1	BOOL	Jos TRUE Q1 asettuu (dominantti)
RESET	BOOL	Jos TRUE Q1 resetoituu
Q1	BOOL	ulostulo

Huom: Kaikki parametrit voidaan negatoida.

Totuustaulu

SET1	RESET	$Q1_{N+1}$
0	0	$Q1_N$
0	1	0
1	0	1
1	1	1



Kuva 13: Esimerkki SR-lohkon käytöstä

4.1.2 RS (Reset Dominant)

Tämä toimilohko toteuttaa RS-kiikun. Jos sisääntulo SET1 = TRUE, ulostulo Q1 asettuu arvoon 1. Q1 pysyy tässä arvossa vaikka SET vaihtuisi arvoon 0. Q1 resetoituu, jos RESET = TRUE. Jos molemmat sisääntulot = TRUE, ulostulo Q1 asettuu RESET:n takia arvoon FALSE.

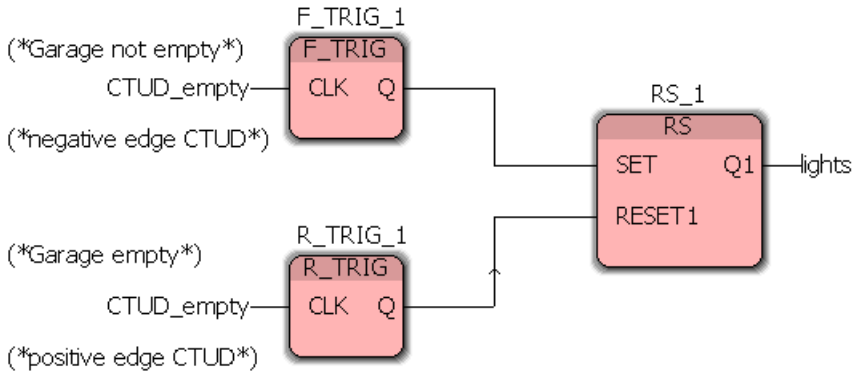
Jos toimilohkoa kutsutaan ensimmäistä kertaa, Q1 = FALSE.

Parametri	Tietotyyppi	Kuvaus
SET	BOOL	Jos TRUE Q1 asettuu
RESET1	BOOL	Jos TRUE Q1 resetoituu (dominantti)
Q1	BOOL	ulostulo

Huom: Kaikki parametrit voidaan negatoida.

Totuustaulu

SET1	RESET	$Q1_{N+1}$
0	0	$Q1_N$
0	1	0
1	0	1
1	1	0



Kuva 14: Esimerkki RS-lohkon käytöstä

4.2 Ajastimet

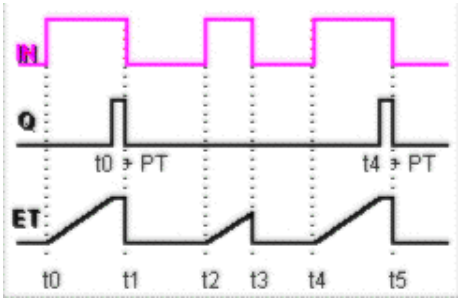
4.2.1 TON (Timer On-Delay)

Tämä ajastintoimilohko viivästyttää päälle laittamista.

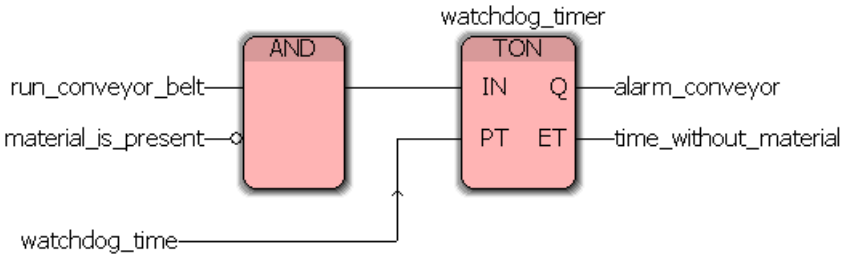
Jos sisääntulo IN vaihtuu arvosta FALSE arvoon TRUE, ulostulon Q vaihtuminen arvoon TRUE viivästyy ajan PT verran. Kun PT on kulunut Q asettuu arvoon TRUE. Ulostulo ET ilmoittaa kuluneen ajan. Vakiomuotoisen ajan sisääntuloon PT voi asettaa esimerkiksi kirjoittamalla muuttujanimeksi t#2s (tarkoittaen kahta sekunttia).

Parametri	Tietotyyppi	Kuvaus
IN	BOOL	Jos havaitaan nouseva reuna, ajastus aloitetaan.
PT	TIME	Viiveen suuruus
Q	BOOL	TRUE jos IN = TRUE ja ET >= PT
		FALSE jos IN = FALSE tai ET < PT
ET	TIME	kulunut aika

Huom: Sisääntulo IN ja ulostulo Q voidaan negatoida.



Kuva 15: TON-lohkon aikakaavio



Kuva 16: Esimerkki TON-lohkon käytöstä

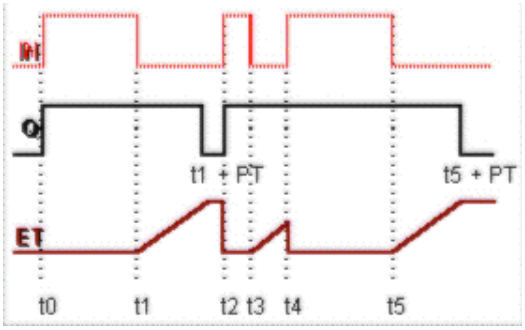
4.2.2 TOF (Timer Off-Delay)

Tämä ajastintoimilohko viivästyttää päältä pois laittamista.

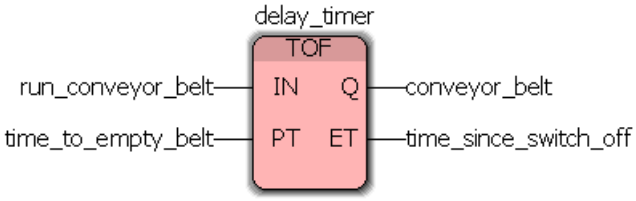
Jos sisääntulo IN vaihtuu arvosta TRUE arvoon FALSE, ulostulon Q vaihtuminen arvoon FALSE viivästyy ajan PT verran. Kun PT on kulunut Q asettuu arvoon FALSE. Ulostulo ET ilmoittaa kuluneen ajan. Vakiomuotoisen ajan sisääntuloon PT voi asettaa esimerkiksi kirjoittamalla muuttujanimeksi t#2s (tarkoittaen kahta sekunttia).

Parametri	Tietotyyppi	Kuvaus
IN	BOOL	Jos havaitaan laskeva reuna, ajastus aloitetaan.
PT	TIME	Viiveen suuruus
Q	BOOL	TRUE jos IN = TRUE ja ET < PT FALSE jos IN = FALSE ja ET >= PT
ET	TIME	kulunut aika

Huom: Sisääntulo IN ja ulostulo Q voidaan negatoida.



Kuva 17: TOF-lohkon aikakaavio



Kuva 18: Esimerkki TOF-lohkon käytöstä

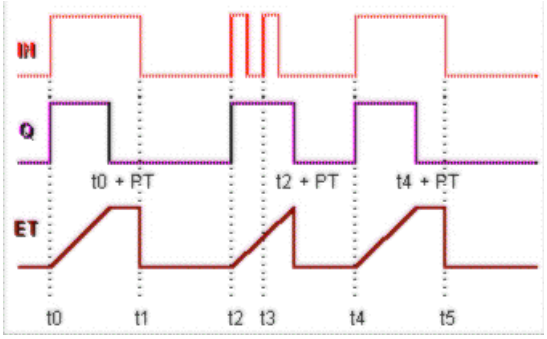
4.2.3 TP (Pulse)

Tämä ajastinlohko tuottaa pulssin.

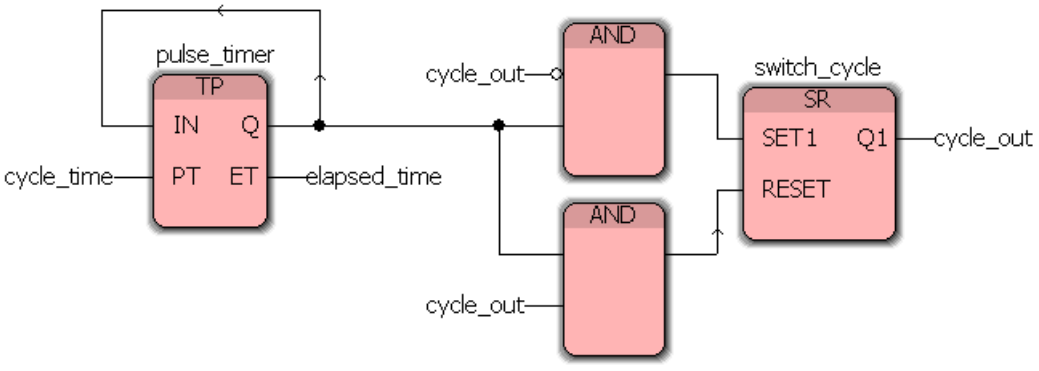
Jos sisääntulo IN vaihtuu arvosta FALSE arvoon TRUE, luodaan ulostuloon Q ajan PT mittainen pulssi. Ulostulo ET kertoo kuluneen ajan. Jos IN vaihtuu arvoon TRUE toisen kerran ja aika PT ei ole vielä kulunut (ET < PT), IN:n arvon muutoksella ei ole vaikutusta ulostuloon Q.

Parametri	Tietotyyppi	Kuvaus
IN	BOOL	Jos havaitaan nouseva reuna, luodaan pulssi.
PT	TIME	pulssin pituus
Q	BOOL	TRUE jos IN = TRUE ja ET < PT FALSE jos IN = FALSE ja ET >= PT
ET	TIME	kulunut aika

Huom: Sisääntulo IN ja ulostulo Q voidaan negatoida.



Kuva 19: TP-lohkon aikakaavio



Kuva 20: Esimerkki TP-lohkon käytöstä

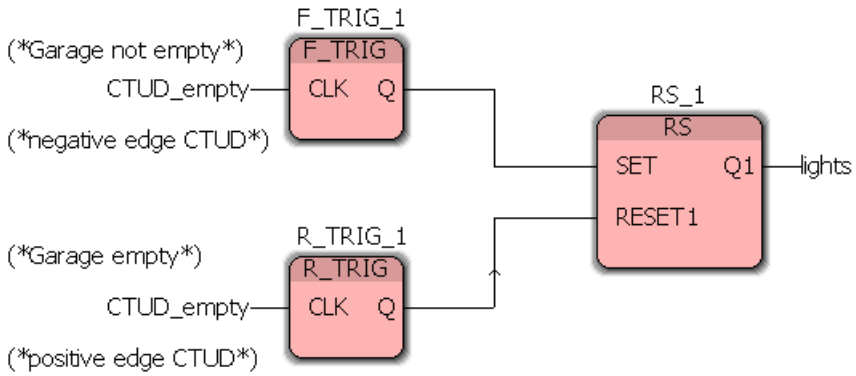
4.3 Liipaisut

4.3.1 R_TRIG (Rising Edge Detection)

Tämä toimilohko havaitsee nousevan reunan (ts. sisääntulon arvon muutoksen 0->1). Jos sisääntulossa CLK havaitaan nouseva reuna, ulostulo Q vaihtuu arvosta FALSE arvoon TRUE. Q pysyy arvossa TRUE toimilohkon suoritukseen asti.

Jos toimilohkoa kutsutaan ensimmäisen kerran, Q = FALSE ensimmäisen reunan havaitsemiseen asti.

Parametri	Tietotyyppi	Kuvaus
CLK	BOOL	Havaitsee nousevan reunan
Q	BOOL	Nousevalla reunalla Q arvosta FALSE arvoon TRUE



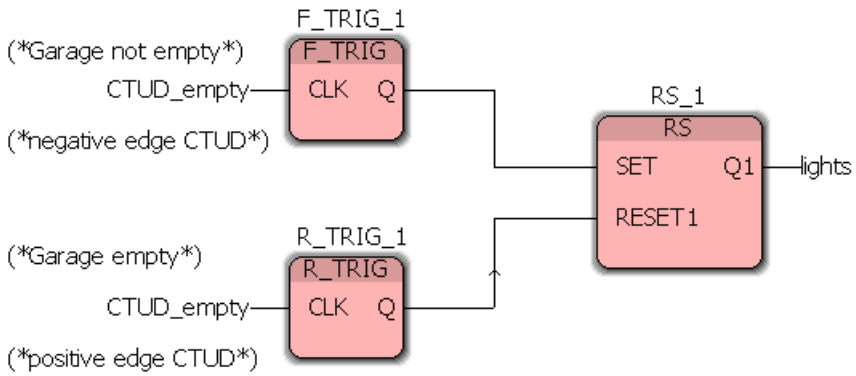
Kuva 21: Esimerkki R_TRIG-lohkon käytöstä

4.3.2 F_TRIG (Falling edge detection)

Tämä toimilohko havaitsee laskevan reunan (ts. sisääntulon arvon muutoksen 1->0). Jos sisääntulossa CLK havaitaan laskeva reuna, ulostulo Q vaihtuu arvosta FALSE arvoon TRUE. Q pysyy arvossa TRUE toimilohkon seuraavaan suoritukseen asti.

Jos toimilohkoa kutsutaan ensimmäisen kerran, Q = FALSE ensimmäisen reunan havaitsemiseen asti.

Parametri	Tietotyyppi	Kuvaus
CLK	BOOL	havaitsee laskevan reunan
Q	BOOL	Laskevalla reunalla Q arvosta TRUE arvoon FALSE



Kuva 22: Esimerkki F_TRIG-lohkon käytöstä

4.4 Laskurit

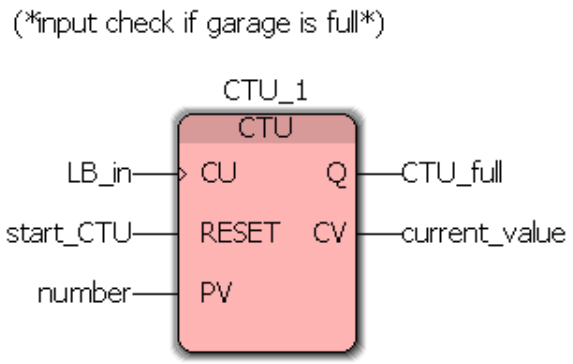
4.4.1 CTU (Counter Up)

Tämä laskuritoimilohko laskee ylöspäin. Jos sisääntulossa CU havaitaan nouseva reuna ja RESET = FALSE, CV:n arvoa lisätään yhdellä. Jos saavutetaan laskurin lopetusarvo (PV), ulostulo Q vaihdetaan arvoon TRUE ja laskenta lopetetaan.

Jos RESET = TRUE, laskuri alustetaan arvolla 0. Laskennan mahdollistamiseksi, sisääntulon RESET arvo pitää olla FALSE. Muuten laskuri alustetaan aina uudestaan.

Parametri	Tietotyyppi	Kuvaus
CU	BOOL	Nousevalla reunalla CV:n arvoa lisätään yhdellä
RESET	BOOL	Jos TRUE, laskuri alustetaan arvolla 0
		Jos FALSE, laskenta mahdollista
PV	INT	ennaltamääritelty arvo
Q	BOOL	TRUE jos CV = PV
CV	INT	laskennan tulos

Huom: Ulostulo Q voidaan negatoida.



Kuva 23: Esimerkki CTU-lohkon käytöstä

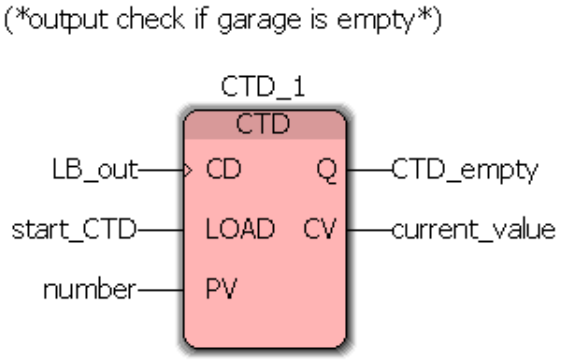
4.4.2 CTD (Counter Down)

Tämä laskuritoimilohko laskee alaspäin. Jos sisääntulossa CD havaitaan nouseva reuna ja LOAD = FALSE, CV:n arvoa vähennetään yhdellä. Jos saavutetaan laskurin lopetusarvo (0), ulostulo Q vaihdetaan arvoon TRUE ja laskenta lopetetaan.

Jos LOAD = TRUE, laskuri alustetaan arvolla PV. Laskennan mahdollistamiseksi sisääntulon LOAD arvo pitää olla FALSE. Muuten laskuri alustetaan aina uudestaan.

Parametri	Tietotyyppi	Kuvaus
CD	BOOL	Nousevalla reunalla CV:n arvoa vähennetään yhdellä
LOAD	BOOL	Jos TRUE, laskuri alustetaan arvolla PV.
		Jos FALSE, laskenta mahdollista
PV	INT	ennaltamääritelty arvo
Q	BOOL	TRUE jos CV = 0
CV	INT	laskennan tulos

Huom: Ulostulo Q voidaan negatoida.



Kuva 24: Esimerkki CTD-lohkon käytöstä

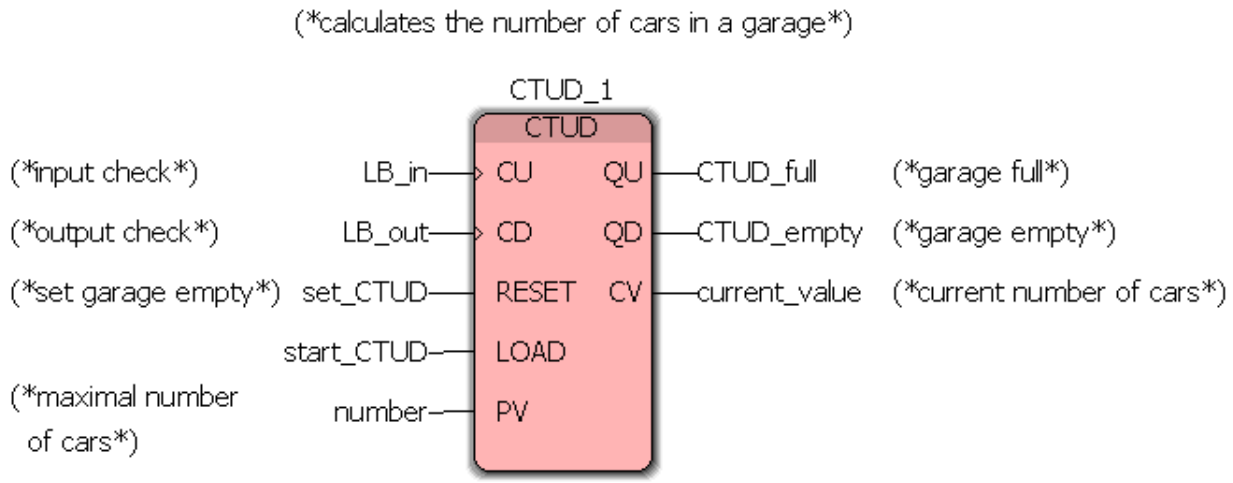
4.4.3 CTUD (Counter Up/Down)

Tämä laskuritoimilohko laskee alas- tai ylöspäin. Jos sisääntulossa CU havaitaan nouseva reuna CV:n arvoa lisätään yhdellä. Jos sisääntulossa CD havaitaan nouseva reuna CV:n arvoa vähennetään yhdellä. Jos CV = PV, ulostulon QU arvoksi asetetaan TRUE. Jos CV = 0, ulostulon QD arvoksi asetetaan TRUE.

Jos RESET = TRUE, laskuri alustetaan arvolla 0. Jos LOAD = TRUE, laskuri alustetaan arvolla PV. Laskennan mahdollistamiseksi sisääntulojen RESET ja LOAD tulee olla FALSE. Muuten laskuri alustetaan aina uudestaan.

Parametri	Tietotyyppi	Kuvaus
CU	BOOL	Nousevalla reunalla CV:n arvoa lisätään yhdellä
CD	BOOL	Nousevalla reunalla CD:n arvoa lisätään yhdellä
RESET	BOOL	Jos TRUE, laskuri alustetaan arvolla 0. Jos FALSE laskenta mahdollista.
LOAD	BOOL	Jos TRUE, laskuri alustetaan arvolla PV. Jos FALSE, laskenta mahdollista.
PV	INT	ennaltamääritelty arvo
QU	BOOL	TRUE jos CV = PV
QD	BOOL	TRUE jos CV = 0
CV	INT	laskennan tulos

Huom: Ulostulot QU ja QD voidaan negatoida.



Kuva 25: Esimerkki CTUD-lohkon käytöstä