BASISSPECIFICATIE

na integratie standaard en speciale versie

verkeerstechnische specificatie van de basisregels van regelingen
V1 31-03-2014

Dit is een uitgave van:

RIJKSWATERSTAAT
Water, Verkeer en Leefomgeving

internetsite: www.rwscregelaar.nl

versie C2014
Maart 2014

Copyright (c) RWS 2003-2014
Alle rechten voorbehouden

INHOUD

INLEIDING

Bij RWS is een filosofie ontwikkeld voor het regelen van verkeer op een met een verkeersregelinstallatie uitgerust kruispunt. Op basis van deze filosofie is een specificatie opgesteld die sinds 1979 zowel binnen als buiten RWS veel gebruikt is voor het ontwerpen en specificeren van regelprogramma's voor verkeersregelinstallaties. Deze specificatie bestaat uit twee onderdelen:
- de *basisspecificatie* en
- de *applicatiespecificatie*.

In de basisspecificatie liggen de op de hierboven bedoelde regelfilosofie gebaseerde basisregels voor regelingen vast. In de applicatiespecificatie moeten de voor een bepaalde regeling unieke gegevens en programmaonderdelen door de ontwerper gespecificeerd worden. Basisspecificatie en applicatiespecificatie samen vormen de specificatie van het programma van een verkeerslichtenregeling.

Enerzijds omdat er al jarenlang behoefte bestond aan een fabrikant onafhankelijke programmeer- en testfaciliteit voor regelprogramma's en anderzijds omdat er inmiddels gestandaardiseerde hogere programmeertalen met bijbehorende compilers beschikbaar waren gekomen die zich leenden voor real-time processen is in 1989 besloten de basisspecificatie te programmeren in de taal C en er een test- en ontwikkelomgeving omheen te bouwen. Voor die test- en ontwikkelomgeving is toen gebruik gemaakt van reeds bestaande onderdelen van FLEXSYT. Mede daardoor ontstond tevens een koppeling met het simulatiedeel van FLEXSYT.

In 1995 is de overstap gemaakt van MS-DOS naar MS-WINDOWS. Daartoe is het programma FLASH ontwikkeld. FLASH is een volledig geintegreerde windows ontwikkel-, test- en simulatieomgeving.
FLASH bevat o.a.
- het simulatiedeel van FLEXSYT II (FLXSIM),
- het regeldeel van FLEXSYT (FLXCOL) en
- een regelaar gebaseerd op de in C geprogrammeerde basisspecificatie.
- een module voor projectbeheer,
- een netwerkeditor,
- een workbench en
- een mogelijkheid tot 2D visualisatie.

Het hierboven bedoelde onderdeel van FLASH dat op de basisspecificatie gebaseerde regelingen kan verzorgen, wordt aangeduid met de naam RWS C-regelaar (CR).

FLASH biedt de mogelijkheid in de kantooromgeving ontworpen regelingen direct te programmeren, te testen, te simuleren en te visualiseren. De relevante onderdelen van een aldus m.b.v. de RWS C-regelaar gespecificeerde regeling kunnen daarna machinaal overgezet worden naar programmatuur die geschikt is om in regeltoestellen te worden geïmplementeerd.

In principe is daarmee de tijdrovende vertaalslag door de fabrikant overbodig geworden en de daarna noodzakelijke afname-procedure bij de fabrikant is belangrijk vereenvoudigd.

Onderdeel van de RWS C-regelaar is de z.g. procesbesturing. Deze procesbesturing verzorgt de afhandeling van de informatiestromen van en naar de regelaar zoals
- detectie-informatie (gaat naar de regelaar) en
- signaalgroepinformatie (komt van de regelaar).

De informatie-uitwisseling tussen het basisspecificatie/applicatiespecificatie-deel en de procesbesturing geschiedt via een (software-software) interface: de z.g. commissie C-interface.

Overzicht van de in de loop der tijd verschenen versies van de basisspecificatie:

1979	eerste uitgave
1980	verbeterde versie
1987 versie 870101	uitbreiding van de vorige versie
1991 versie C910301	aanpassing t.b.v. de programmering in C
1991 november	aanvulling
1992 versie C921111	versie C910301 met aanvulling van november 1991
2003 versie C2003	versie C921111 met enkele kleine wijzigingen die in de jaren 1992-2003 in de RWS C-regelaar zijn aangebracht
2005 versie C2005	versie C2003 met enkele kleine wijzigingen die in de jaren 2003-2005 in de RWS C-regelaar zijn aangebracht
2014 versie C2014	versie na integratie CR- en CRS-versie 2005 waarin bezet- en hiaattijden per detector, speciale realisatie, retour-wachtgroen, VLOG en KSU/KSH-toewijzing bij uitgaande koppelsignalen zijn toegevoegd

Er is een toelichting op deze basisspecificatie beschikbaar waarin een algemene beschrijving van de hierboven genoemde filosofie is opgenomen en waarin gedetailleerde uitleg over elke in de basisspecificatie gebruikte formule wordt gegeven.

Voor gedetailleerde informatie over de RWS C-regelaar wordt verwezen naar de handleiding van deze regelaar.

Voor de RWS C-regelaar zijn er standaarden en voorbeelden beschikbaar voor ontwerpers.
Standaarden zijn stukken programmatuur (functies) waarin veel voorkomende zaken op een standaard manier zijn opgelost en die de ontwerper in zijn regeling kan integreren.
Voorbeelden zijn regelingen waarin oplossingen voor bepaalde problemen zijn uitgewerkt. Een ontwerper kan deze oplossingen overnemen.
De standaarden en de voorbeelden zijn beschikbaar op de internetsite: www.rwscregelaar.nl.

HOOFDSTUK 1 Algemene voorwaarden

a) Deze basisspecificatie geeft tezamen met de applicatiespecificatie de verkeerstechnische beschrijving van de door de verkeerslantaarns van de verkeersregelinstallatie te tonen beelden op de te regelen kruising.

b) Met in achtname van het onder a gestelde dient aan deze basisspecificatie voldaan te worden tenzij in de applicatiespecificatie anders is vermeld.

c) Alle in de applicatiespecificatie genoemde signaalgroepen, detectoren, drukknoppen, tijden e.d. dienen in het regelprogramma aanwezig te zijn.

HOOFDSTUK 2 Legende

2.1 Afkortingen en overige aanduidingen

De in dit hoofdstuk gespecificeerde schrijfwijze (incl. die van de indices) is verplicht.

Bij de indices staat

i	voor een volgnummer
j	voor een algemene index
k	voor een algemene index
l	voor een bloknummer
m	voor het bloknummer van het actieve blok
m⊕b	voor een bloknummer van een niet actief blok
n	voor een signaalgroepnummer

Verder betekent:

dg	een dag van de week zoals ZONDAG of MAANDAG enz.
4’	een 4 cijferig getal
2’	een 1 of 2 cijferig getal
x	een getal tussen 1 en 32767

Verder geldt dat

- indices worden geschreven in arabische cijfers en worden gescheiden door een "," of een "_".
- "naam" staat voor een vrij te kiezen afkorting of aanduiding onder de restrictie dat "naam"
  - tezamen met de eventueel daaraan toe te voegen index of indices uniek dient te zijn,
  - niet gelijk mag zijn aan een andere binnen de RWS C-regelaar gebruikte afkorting,
  - niet gelijk mag zijn aan een keyword van C en
  - maximaal 20 karakters mag bedragen.

A(SGn)	aanvraag SGn
AAN	aan
AFK(SGn)	afkappen SGn
AFK_OP(SGn)	afkapoptie SGn
ALT	alternatief
AR(SGn, BKl)	alternatieve realisatie SGn van BKl
AVR	aanvraag
BEURT	beurt (in regeling zonder blokken)
BKl	blok l
BL(SGn)	blokkeren SGn
BL_OP(SGn)	blokkeeroptie SGn

BLOK(BKl)	variabele blok van BKl
BO(BKl)	tegenhouden overgang BKl
BO_OP(BKl)	optie m.b.t. BO(BKl)
BR(SGn)	bijzondere realisatie SGn
BS_H5p	basisspecificatie hoofdstuk 5p
D(SGn_i)	melding van detector SGn_i
DA(SGn)	detectoraanvraag SGn
dag(dg)	weekdag zoals maandag, dinsdag enz.
dagnummer()	nummer van de kalenderdag
datum(4', 2', 2')	kalenderdatum in jaar, maand en dag
DFA	bij DFOUT continue melding
DFOUT(SGn_i)	detectiebewaking voor D(SGn_i) aangesproken
DFOUT_BG(SGn_i)	detectiebewaking voor D(SGn_i) aangesproken op bovengedrag
DFOUT_OG(SGn_i)	detectiebewaking voor D(SGn_i) aangesproken op ondergedrag
DFU	bij DFOUT geen melding
DK	drukknop
DP	aanwezigheidsdetector
DS	selectieve detector
EGGPARM	extra geheel getal parameter
EPARM	extra parameter
FB	fasebewaking
FIX	fixatie
G(SGn)	groen/wit SGn
GEEN	geen specifieke betekenis
GEEN_blk_g(SGn)	formule g (hoofdstuk 6) is voor SGn niet werkzaam
GG_TIJD(SGn)	garantiegroentijd SGn
GGL_TIJD(SGn)	garantiegeel-/groenknippertijd SGn
GL(SGn)	geel/groenknipperen SGn
GL_TIJD(SGn)	geel/groenknippertijd SGn
GO_TIJD(SGn, SGj)	garantie-ontruimingstijd van SGn naar SGj
GR_TIJD(SGn)	garantieroodtijd SGn
H1e	1e hiaat
H_TIJD(SGn_i)	hiaattijd D(SGn_i)
H_TIJDi(SGn_i)	hiaattijd i D(SGn_i)
H2e	2e hiaat
HF(naam)	hulpvariabele naam, ook wel hulpfunctie genoemd
INT	interface tussen procesbesturing en regelaar
jaarnummer()	nummer van het kalenderjaar
KL(naam)	klok naam
KSU	Uitgaand koppelsignaal door procesbesturing laten resetten als verzendende regelaar niet regelt

KSH	Besturing uitgaand koppelsignaal door procesbesturing laten handhaven als verzendende regelaar niet regelt
maandnummer()	nummer van de kalendermaand
MA(SGn)	meeaanvraag SGn
MA_OP(SGn)	meeaanvraagoptie SGn
MAH4(SGn, i)	meeaanvraag hulpvariabele 4 SGn en detectorpaar i
MAH5(SGn, i)	meeaanvraag hulpvariabele 5 SGn en detectorpaar i
MAH6(SGn, i)	meeaanvraag hulpvariabele 6 SGn en detectorpaar i
MAH7(SGn, i)	meeaanvraag hulpvariabele 7 SGn en detectorpaar i
MG_TIJD(SGn)	maximum groentijd SGn
MG_TIJDi	set maximum groentijden i
MVG(SGn)	meeverlenggroen SGn
NG	niet gebruikt
NUMBK	aantal blokken
NUMD	aantal detectoren
NUMEP	aantal extra parameters
NUMHF	aantal hulpfuncties
NUMKL	aantal klokken
NUMOV	aantal overige symbolische namen
NUM_OVI	aantal overige ingangssignalen
NUM_OVU	aantal overige uitgangssignalen
NUMSCH	aantal schakelaars
NUMSG	aantal signaalgroepen
NUMT	aantal tijden
NUMTL	aantal tellers
O_TIJD(SGn, SGj)	ontruimingstijd van SGn naar SGj
PG(SGn, BKl)	primair gerealiseerd SGn van BKl
PPA(SGn, BKl)	privilege periode alternatief SGn van BKl
PPA_OP(SGn, BKl)	optie m.b.t. PPA(SGn, BKl)
PPAH(SGn)	PPA hulpvariabele SGn
PPB(SGn)	privilege periode bijzonder SGn
PPB_OP(SGn)	optie m.b.t. PPB(SGn)
PPP(SGn, BKl)	privilege periode primair SGn van BKl
PPP_OP(SGn, BKl)	optie m.b.t. PPP(SGn, BKl)
PPS(SGn)	privilege periode speciaal SGn
PPS_OP(SGn)	optie m.b.t. PPS(SGn)
PPV(SGn)	privilege periode vooruitschakelen SGn
PPV_OP(SGn)	optie m.b.t. PPV(SGn)
PR(SGn, BKl)	primaire realisatie SGn van BKl
PRIM	primair
Q	tijdsinstelling volgens tijdentabel

R(SGn)	rood SGn
RGAH(SGn, j)	richtinggevoelige-aanvraag hulpvariabele SGn en detectorpaar j
ROG(SGn)	recht op groen SGn
RVAG2e(SGn)	retour 2e voertuigafhankelijkgroen (SGn)
RVAG2e_OP(SGn)	optie m.b.t. RVAG(SGn)
RVG(SGn)	rood voor groen SGn
RWG_OP(SGn)	Optie m.b.t. terugkeren naar wachtgroen (WG)
RWR(SGn)	retour wachtrood SGn
RWR_OP(SGn)	optie m.b.t. RWR(SGn)
SCH(naam)	schakelaar naam
SGn	signaalgroep n
status_regelen(x)	regelstatus van regeling x
STOP	einde gegevens
tijd(2', 2')	werkelijke tijd in uren en minuten
TIJD(naam)	instelbare tijd naam
TL(naam)	teller naam
TMVG(SGn)	toestaan meeverlenggroen SGn
TMVG_OP(SGn)	optie m.b.t. toestaan meeverlenggroenoptie SGn
TO_PRIM(SGn)	tegenhouden van de overnames van signaalgroep n naar primair
TOEPASSEN	toepassen
TR(SGn)	tegenhouden realisatie SGn
TR_OP(SGn)	optie m.b.t. TR(SGn)
UIT	uit
VAA(SGn)	versneld afsluiten van het aanvraaggebied SGn
VAA_OP(SGn)	optie m.b.t. VAA(SGn)
VAG1e(SGn)	1e voertuigafhankelijk groen SGn
VAG2e(SGn)	2e voertuigafhankelijk groen SGn
voer_dump_uit()	maak een dump aan
VG(SGn)	vastgroen SGn
VG_TIJD(SGn)	vastgroentijd SGn
VMVG(SGn)	vasthouden MVG(SGn)
VMVG_OP(SGn)	optie m.b.t. VMVG_OP(SGn)
VNM(SGn)	versneld naar meeverlenggroen of wachtrood (voor een alternatief gerealiseerde) SGn
VNM_OP(SGn, BKl)	optie m.b.t. VNM(SGn)
VNMH(SGn)	versneld naar MVG hulpvariabele SGn
VRVG(SGn)	vasthouden RVG(SGn)
VRVG_OP(SGn)	optie m.b.t. vasthouden RVG SGn
VVAG2e(SGn)	vasthouden VAG2e(SGn)
VVAG2e_OP(SGn)	optie m.b.t. vasthouden VAG2e(SGn)
W_REG_VRI	werkelijk regelen van de verkeersregelinstallatie

WAAR	altijd waar
WB	wachtblok
WG(SGn)	wachtgroen SGn
WR(SGn)	wachtrood SGn
WSGR	wachtstandgroen
WSRH	Wachtstandrood hulpvariabele SGn

2.2 Gebruikte conventies

De volgende conventie zijn gebruikt:

notatie:	betekenis:
||	logisch "of"
&&	logisch "en"
!	logisch "niet"
=	toekennings-operator

Deze betekenis is dezelfde als die van de programmeertaal C.
Commentaar staat tussen /* en */ of achter //.

Indien in een voorwaarde gerefereerd wordt aan signaalgroepen, die op het moment van evaluatie van deze voorwaarde niet gedefinieerd zijn, is de beschouwde formule per definitie waar, tenzij anders is aangegeven.

2.3 Variabelen, parameters, instructies, referenties en toewijzingen

De basisspecificatie en de applicatiespecificatie kennen
- variabelen,
- parameters,
- instructies,
- referenties en
- toewijzingen.

De basis van alle gebruikte namen voor variabelen, parameters, instructies en referenties en enkele toewijzingen is vastgelegd in 2.1.

De eigenschappen van de belangrijkste **variabelen** worden beschreven in hoofdstuk 4.

Een **parameter** is een variabele die door de gebruiker on-line (tijdens regelen) van waarde kan worden veranderd (ingesteld). De basisinstelling wordt vastgelegd in de applicatiespecificatie.

Instructies worden binnen de basisspecificatie en de applicatiespecificatie gebruikt om de regeling te programmeren. De naamgeving van instructies is afgeleid van de bijbehorende variabele, parameter of toewijzing door er het volgende aan vooraf te laten gaan:

act_	incr_	set_
decr_	kies_	start_
dehalteer_	maak_	wijzig_
halteer_	plaats_in_	
herstart_	reset_	

en bij enkele er tevens het volgende aan toe te voegen:

()	_toewijzing(SGn)
_toewijzing	_toewijzing_D(SGn_i)

Lang niet alle voorvoegsels en toevoegingen kunnen worden gecombineerd met alle afkortingen uit 2.1. In de handleiding van de RWS C-regelaar staat een limitatieve opsomming van alle instructies die binnen de RWS C-regelaar voor de applicatieschrijver beschikbaar zijn.

Referenties worden binnen de basisspecificatie en de RWS C-regelaar gebruikt bij het formuleren van instructies. Aan vrijwel elke variabele kan worden gerefereerd. De naamgeving van de overige referenties is afgeleid van de bijbehorende variabele of parameter door er het volgende aan vooraf te laten gaan:

begin_	waarde_	nw_info_in_
einde_	halterende_	
inst_	haal_uit_	

Lang niet alle voorvoegsels kunnen worden gecombineerd met alle afkortingen uit 2.1. In de handleiding van de RWS C-regelaar staat een limitatieve opsomming van alle referenties die binnen de RWS C-regelaar voor de applicatieschrijver beschikbaar zijn.

Toewijzingen worden in de applicatiespecificatie in tabellen gebruikt om bepaalde eigenschappen te kunnen toekennen. In deze basisspecificatie worden alleen die toewijzingen genoemd waarvan de naam als basis dient voor een bepaalde instructie. Voor een compleet overzicht van alle mogelijke toewijzingen zie de handleiding van de RWS C-regelaar.

HOOFDSTUK 3 Verklaring van enkele begrippen

Applicatiespecificatie	Door de ontwerper te definiëren voorwaarden voor een specifieke regeling.
Basisspecificatie	Basisregels van de basisstuctuur.
Beurt	Hulpvariabele waarmee de overgang WR naar ROG kan worden bewerkstelligd in regeling zonder blokken.
Blok	Verzameling van richtingen; BLOK(BKl) komt tevens als variabele voor.
Conflictrichting	Richting van een verkeersstroom die niet gelijktijdig met een beschouwde verkeersstroom op de kruising wordt toegelaten.
Detector	Element of samenstel van elementen met behulp waarvan verkeersmeldingen aan het verkeersregeltoestel worden doorgegeven.
Drukknop	Apparaat, dienend om een verkeersmelding met de hand tot stand te brengen.
Fasecyclus	Opeenvolging van een groen (wit)-, geel (groen-knipper)- en roodfase per verkeerslantaarn.
Fictief conflict	In de applicatiespecificatie te definiëren conflict dat in de structuur van de regeling als zodanig wordt beschouwd bij de overgangen zoals die gespecificeerd zijn bij de blokprocedure. Hier ook gebruikt voor: Verzameling van toestanden die per signaalgroep kunnen optreden.
Optie	In de applicatiespecificatie te specificeren variabele door middel waarvan de afwikkeling van het regelprogramma kan worden beïnvloed.
Overgang	Toestandsverandering
Procesbesturing	Verzorgt de afhandeling van de informatiestroom van en naar de buitenwereld
Richting	Zie signaalgroep
Signaalgroep	Groep verkeerslantaarns die eenzelfde fasecyclus hebben.
Stuurvoorwaarde	Voorwaarde ter bepaling van overgangen.
Teller	Geheugenelement voor numerieke waarden.
Tijd	Instelbaar tijdelement ter bepaling van de lengte van een periode.
Toestand	Deel van de fasecyclus van een signaalgroep.
Variabele	Deel van het regelprogramma met specifieke eigenschappen.
Wachtstandrichting	Signaalgroep die in wachtgroen wacht indien geen aanvragen van de conflictrichtingen aanwezig zijn.

HOOFDSTUK 4 Eigenschappen

⇒ Blok

Een in het blokkenschema in een bepaald blok als primair c.q. alternatief opgegeven signaalgroep is een bij dat blok behorende primaire c.q. alternatieve signaalgroep.

⇒ Detectorfouten

De procesbesturing geeft als volgt informatie over de status van de detectoren:

DFOUT(SGn_i)	detector (SGn_i) stoort
DFOUT_OG(SGn_i)	detector (SGn_i) stoort op ondergedrag
DFOUT_BG(SGn_i)	detector (SGn_i) stoort op bovengedrag

⇒ Extra geheel getal parameter

Parameter die niet als variabele beschikbaar is.
Een extra geheel getal parameter krijgt de waarde x na de instructie:

wijzig_EGGPARM(naam, x).

Aan de instelling van een extra heel getal parameter kan worden gerefereerd:
inst_EGGPARM(naam)>x (> geldt als voorbeeld; alle relationele operatoren van de taal C zijn toegestaan).

wijzig_EGGPARM(naam, x)	kan worden opgegeven als instructies die moeten worden uitgevoerd

Aan inst_EGGPARM(naam) > x kan worden gerefereerd.

Extra geheel getal parameters dienen in de RWS C-regelaar als een geheel getal te worden opgegeven. Dit betekent b.v. dat een instelling van "drie" moet worden geschreven als "3".

⇒ **Extra parameter**

Parameter die niet als variabele beschikbaar is.
Een extra parameter krijgt de waarde x na de instructie: wijzig_EPARM(naam, x).
Aan de instelling van een extra parameter kan worden gerefereerd.: inst_EPARM(naam)>x (> geldt als voorbeeld; alle relationele operatoren van de taal C zijn toegestaan).

wijzig_EPARM(naam, x) kan worden opgegeven als instructies die moeten worden uitgevoerd

Aan inst_EPARM(naam) > x kan worden gerefereerd.

Extra parameters dienen in de RWS C-regelaar als float te worden opgegeven. Dit betekent b.v. dat een instelling van "drie" moet worden geschreven als "3.0".

⇒ **Fasebewaking**

Een fasebewaking heeft tot gevolg dat een dump wordt aangemaakt en dat fasebewaking wordt aangeboden aan de procesbesturing.

⇒ **Fixatie**

Fixatie wordt gecreëerd door de procesbesturing en wordt aan de regelaar aangeboden.

⇒ **Klok**

Een klok is een variabele die waar is gedurende de in de applicatiespecificatie op te geven perioden en niet waar is buiten deze perioden. De tijdstippen die deze perioden bepalen zijn als parameter van het regelprogramma instelbaar.

⇒ **Optie**

Een niet gespecificeerde optie is per definitie niet waar.

De basisspecificatie kent de volgende opties:

AFK_OP(SGn)	PPS_OP(SGn)	TR_OP(SGn)
BL_OP(SGn)	PPV_OP(SGn)	VAA_OP(SGn)
BO_OP(BKl)	RVAG2e_OP(SGn)	VRVG_OP(SGn)
MA_OP(SGn)	RWG_OP(SGn)	VVAG2e_OP(SGn)
PPA_OP(SGn, BKl)	RWR_OP(SGn)	VMVG_OP(SGn)
PPB_OP(SGn)	TMVG_OP(SGn)	VNM_OP(SGn, BKl)
PPP_OP(SGn, BKl)		

⇒ **Schakelaar**

Een schakelaar is een variabele die als parameter van het regelprogramma instelbaar is.

⇒ **Signaalgroep**

Als G(SGn) waar is, toont de signaalgroep groen (voor niet-openbaar vervoersrichtingen) of wit (voor openbaar vervoersrichtingen).
Als GL(SGn) waar is, toont de signaalgroep geel (voor niet-voetgangersrichtingen) of groenknipperen (voor voetgangersrichtingen).
Als R(SGn) waar is, toont de signaalgroep rood.
Een in het blokkenschema als zodanig opgegeven signaalgroep is een wachtstandrichting.
Een signaalgroep kan als primaire en als alternatieve signaalgroep zijn gespecificeerd.
Primaire signaalgroepen van één blok zijn onderling niet conflicterend.

⇒ **Teller**

Een teller kan waar zijn (tellerinhoud > 0): TL(SGn_i).
Een teller kan niet waar zijn (tellerinhoud is 0): !TL(SGn_i).
Een teller wordt met één opgehoogd na de instructie: incr_TL(SGn_i).
Een teller wordt met één verlaagd na de instructie: decr_TL(SGn_i).
Een teller krijgt de waarde x na de instructie: maak_TL(SGn_i, x).
Aan de waarde van een teller kan worden gerefereerd: TL(SGn_i)>x (> geldt als voorbeeld; alle relationele operatoren van de taal C zijn toegestaan).

incr_TL(SGn_i)
decr_TL(SGn_i)
en maak_TL(SGn_i, x) kunnen worden opgegeven als instructies die moeten worden uitgevoerd.

Aan TL(SGn_i)
!TL(SGn_i)
(TL(SGn_i)>x) kan worden gerefereerd.
(> geldt als voorbeeld)

⇒ **Tijd**

Een tijd kan lopen: TIJD(naam).
Een tijd kan niet lopen (afgelopen zijn): !TIJD(naam).
Het lopen van een tijd kan gehalteerd zijn: halterende_TIJD(naam). Tijdens halteren blijft de variabele TIJD(naam) waar.
Een tijd gaat vanaf de waarde 0 lopen na de instructie: start_TIJD(naam). Een dergelijke startinstructie heeft echter geen uitwerking tijdens het lopen of het halteren van een tijd.
Een tijd gaat (opnieuw) vanaf de waarde 0 lopen na de instructie: herstart_TIJD(naam).
Een tijd is afgelopen (loopt niet) na het overschrijden van de ingestelde waarde Q of na de instructie: reset_TIJD(naam).
Een lopende tijd wordt gehalteerd na de instructie: halteer_TIJD(naam).
Het halteren van een tijd wordt beëindigd
- en de tijd gaat verder met lopen na de instructie: dehalteer_TIJD(naam),
- en de tijd gaat opnieuw vanaf de waarde 0 lopen na de instructie: herstart_TIJD(naam),
- en het aflopen van de tijd wordt beëindigd na de instructie: reset_TIJD(naam).
De overgang van niet-lopen naar lopen van een tijd wordt aangeduid met: begin_TIJD(naam).
De overgang van lopen naar niet-lopen van een tijd wordt aangeduid met: einde_TIJD(naam).

Aan de instelling van een tijd kan worden gerefereerd: inst_TIJD(naam) > x (> geldt als voorbeeld; alle relationele operatoren van de taal C zijn toegestaan).
Aan de momentane waarde van een lopende tijd kan worden gerefereerd: waarde_TIJD(naam) > x (> geldt als voorbeeld; alle relationele operatoren van de taal C zijn toegestaan).
De instelling van een tijd kan worden gewijzigd: wijzig_TIJD(naam, x).

start_TIJD(naam)
herstart_TIJD(naam)
halteer_TIJD(naam)
dehalteer_TIJD(naam)
wijzig_TIJD(naam, x)
en reset_TIJD(naam) kunnen worden opgegeven als instructies die moeten worden uitgevoerd.

Aan TIJD(naam)
halterende_TIJD(naam)
!TIJD(naam)
inst_TIJD(naam)>x
waarde_TIJD(naam)>x
begin_TIJD(naam)
en einde_TIJD(naam) kan worden gerefereerd
(> geldt als voorbeeld)

De hierboven gebruikte TIJD(naam) dient als voorbeeld.

Niet alle referenties en instructies van bepaalde tijden zijn voor de applicatieschrijver beschikbaar (b.v. start_O_TIJD(SGn, SGj)). In de handleiding van de RWS C-regelaar staat een limitatieve opsomming van alle referenties en instructies die binnen de RWS C-regelaar voor de applicatieschrijver beschikbaar zijn.
Alle tijden dienen in de RWS C-regelaar als float te worden opgegeven. Dit betekent b.v. dat een tijdsinstelling van "drie" seconden moet worden geschreven als "3.0" seconden.
Instellingen kunnen echter niet negatief zijn.

De basisspecificatie kent de volgende tijden:

GG_TIJD(SGn)	GR_TIJD(SGn)	O_TIJD(SGn, SGj)
GGL_TIJD(SGn)	H_TIJD(SGn_i)	TIJD(SGn_i)
GL_TIJD(SGn)	B_TIJD(SGn_i)	VG_TIJD(SGn)
GO_TIJD(SGn, SGj)	MG_TIJD(SGn)	

⇒ **Toestand**

De basisspecificatie kent de volgende toestanden:

G(SGn)	ROG(SGn)	VG(SGn)
GL(SGn)	RVG(SGn)	WG(SGn)
MVG(SGn)	VAG1e(SGn)	WR(SGn)
R(SGn)	VAG2e(SGn)	

⇒ **Variabele**

Een variabele kan waar (actief) zijn: HF(naam).
Een variabele kan niet waar (niet actief) zijn: !HF(naam).
Een variabele wordt waar na de instructie: set_HF(naam) (geldt niet voor D(SGn)).
Een variabele wordt niet waar na de instructie: reset_HF(naam) (geldt niet voor D(SGn)).
De overgang van niet waar naar waar van een variabele wordt aangeduid met: begin_HF(naam).

De overgang van waar naar niet waar van een variabele wordt aangeduid met: einde_HF(naam).

set_HF(naam)
en reset_HF(naam) kunnen worden opgegeven als instructies die moeten worden uitgevoerd.

Aan HF(naam)
!HF(naam)
begin_HF(naam)
en einde_HF(naam) kan worden gerefereerd.

De hierboven gebruikte HF(naam) dient als voorbeeld.

Tellers en tijden hebben afwijkende eigenschappen.
Niet alle referenties en instructies van bepaalde variabelen zijn voor de applicatieschrijver beschikbaar (b.v. set_G(SGn)). In de handleiding van de RWS C-regelaar staat een limitatieve opsomming van alle referenties en instructies die binnen de RWS C-regelaar voor de applicatieschrijver beschikbaar zijn.

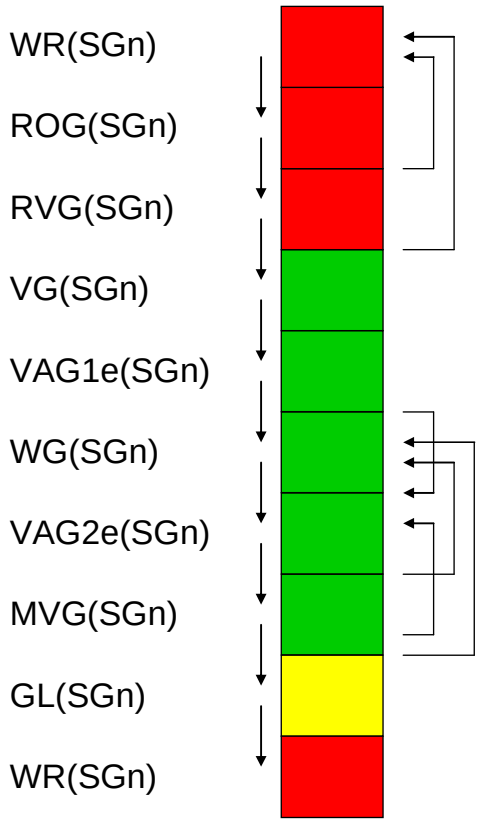
De basisspecificatie kent de volgende variabelen:

A(SGn)
AFK(SGn)
AFK_OP(SGn)
AR(SGn, BKl)
BEURT(SGn)
BL(SGn)
BLOK(BKl)
BL_OP(SGn)
BO(BKl)
BO_OP(BKl)
BR(SGn)
B_TIJD(SGn_i)
D(SGn_i)
DA(SGn)
DFOUT(SGn_i)
DFOUT_BG(SGn_i)
DFOUT_OG(SGn_i)
FB
FIX
G(SGn)
GEEN_blk_g(SGn)
GG_TIJD(SGn)
GGL_TIJD(SGn)
GL(SGn)
GL_TIJD(SGn)
GO_TIJD(SGn, SGj)
GR_TIJD(SGn)
HF(naam)
H_TIJD(SGn_i)
H1e_loopt(SGn)
H2e_loopt(SGn)
INT
KL(SGn_i)
MA(SGn)
MA_OP(SGn)
MAH4(SGn, i)
MAH5(SGn, i)
MAH6(SGn, i)
MAH7(SGn, i)
MG_TIJD(SGn)
MVG(SGn)
O_TIJD(SGn, SGj)
PG(SGn, BKl)
PPA(SGn, BKl)
PPAH(SGn)
PPA_OP(SGn, BKl)
PPB(SGn)
PPB_OP(SGn)
PPP(SGn, BKl)
PPP_OP(SGn, BKl)
PPS(SGn)
PPS_OP(SGn)
PPV(SGn)
PPV_OP(SGn)
PR(SGn, BKl)
R(SGn)
RGAH(SGn, j)
ROG(SGn)
RVAG2e(SGn)
RVAG2e_OP(SGn)
SCH(SGn_i)
RWG_OP(SGn)
RWR(SGn)
RWR_OP(SGn)
TIJD(naam)
TL(naam)
TO_PRIM(SGn)
TMVG(SGn)
TMVG_OP(SGn)
TR(SGn)
TR_OP(SGn)
VAA(SGn)
VAA_OP(SGn)
VAG1e(SGn)
VAG2e(SGn)
VG_TIJD(SGn)
VNM(SGn)
VNMH(SGn)
VNM_OP(SGn, BKl)
VMVG(SGn)
VMVG_OP(SGn)
VVAG2e(SGn)
VVAG2e_OP(SGn)
VRVG(SGn)
VRVG_OP(SGn) VG(SGn)
WG(SGn)
WR(SGn)
WSRH
W_REG_VRI

HOOFDSTUK 5 Afwikkeling van de fasecyclus

5.1 Algemeen

Een fasecyclus bevat de volgende toestanden:



In de bovenstaande figuur geven de getekende pijltjes de mogelijke overgangen aan.
Voor niet-wachtstandrichtingen geldt per definitie dat WG(SGn) niet wordt doorlopen.

De overgangen van WR(SGn) naar ROG(SGn) zijn beschreven bij de blokprocedure (hoofdstuk 6). De overgang van ROG(SGn) naar WR(SGn) kan zowel plaatsvinden in de signaalgroepprocedure (zie hoofdstuk 5) als in de blokprocedure (zie hoofdstuk 6).

Alle overige overgangen zijn beschreven bij de signaalgroepprocedure (hoofdstuk 5). Hierbij wordt onderscheid gemaakt in regelingen met blokkenschema (paragraaf 5.2) en regelingen zonder blokkenschema (paragraaf 5.3).

5.2 Overgangsformules in regeling met blokkenschema

De hierna gebruikte afkortingen a, b, c, d, ... enz. gelden slechts binnen hoofdstuk 5.

⇒ **van WR naar ROG**

```
/*  voor reset_WR(SGn) en set_ROG(SGn) zie blokprocedure */
```

⇒ **van ROG naar RVG**

```
reset_ROG(SGn)
en set_RVG(SGn)       als    a  && !(c || si)
                      waar is

        a = ROG(SGn) && A(SGn)
        c =    VNM(SGn) || BL(SGn)
                  /*  voor VNM(SGn)    zie 6.5.4      */
                  /*  en voor BL(SGn)  zie 5.4     */
        si = RWR(SGn)

        /*  zie voor reset_ROG(SGn) ook blokprocedure   */
        /*  voor A(SGn)  zie 5.4                        */
```

⇒ **einde ROG**

```
reset_ROG(SGn)    als    c || si
                  waar is

        c =    VNM(SGn) || BL(SGn)
                  /*  voor VNM(SGn)    zie 6.5.4      */
                  /*  en voor BL(SGn)  zie 5.4     */
        si = RWR(SGn)
```

⇒ **einde RVG**

reset_RVG(SGn) als b && (c || si || !c && d && e && av && !aw)
waar is

b = RVG(SGn)
c = VNM(SGn) || BL(SGn)
/* voor VNM(SGn) zie 6.5.4 */
/* en voor BL(SGn) zie 5.4 */
si = RWR(SGn)
/* let op: op basis van RVG(SGn) kan meeverlenggroen */
/* van andere signaalgroepen zijn beëindigd */
d = voor alle in de conflictmatrix van de applicatiespecificatie gedefinieerde conflictrichtingen j van SGn is R(SGj) waar
e = voor alle ontruimingstijden naar SGn is !O_TIJD(SGj, SGn) waar
av = voor alle garantie-ontruimingstijden naar SGn is !GO_TIJD(SGj, SGn) waar
aw = VRVG(SGn)

⇒ **van RVG naar VG**

reset_R(SGn)
set_G(SGn)
en set_VG(SGn) als b && !c && d && e && av && !aw
waar is

b = RVG(SGn)
c = VNM(SGn) || BL(SGn)
/* voor VNM(SGn) zie 6.5.4 */
/* en voor BL(SGn) zie 5.4 */ d = voor alle in de conflictmatrix van de applicatiespecificatie gedefinieerde conflictrichtingen j van SGn is R(SGj) waar
e = voor alle ontruimingstijden naar SGn is !O_TIJD(SGj, SGn) waar
av = voor alle garantie-ontruimingstijden naar SGn is !GO_TIJD(SGj, SGn) waar
aw = VRVG(SGn)

⇒ **van VG naar VAG1e**

reset_VG(SGn)
en set_VAG1e(SGn)
als af && (ag || ah && ak)
waar is

af = VG(SGn)
ag = !VG_TIJD(SGn) || AFK(SGn)
/* voor AFK(SGn) zie 5.4 */
ah = VNM(SGn)
ak = voor een met SGn conflicterende SGj is RVG(SGj) waar

⇒ **einde VAG1e**

reset_VAG1e(SGn) als h && (o || p || q || ah && ak)
waar is

h = VAG1e(SGn)
o = er is geen detector aan SGn toegewezen met een H1e_functie of er is geen hiaattijd voor detectoren met een H1e_functie die aan SGn zijn toegewezen gedefinieerd
p = !H1e_loopt(SGn) && !D(SGn_i) || AFK(SGn)
in de bovenstaande formule staat H1e_loopt(SGn) voor het lopen van een hiaattijd van een detector met een H1e_toewijzing die is toegewezen aan SGn. SGn_i staat voor de detectoren waaraan een H1e-functie is toegewezen
q = !MG_TIJD(SGn)
ah = VNM(SGn)
ak = voor een met SGn conflicterende SGj is RVG(SGj) waar

⇒ **van VAG1e naar WG**

set_WG(SGn) als r && s && t
waar is

r = einde_VAG1e(SGn)
s = SGn is een wachtstandrichting
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar

⇒ **einde_WG**

reset_WG(SGn) als u && (!t || v)
waar is

u = WG(SGn)
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar
v = AFK(SGn)

⇒ **begin VAG2e**

set_VAG2e(SGn) als r && (!s || !t) || u && (!t || v) || ba && sj
waar is

r = einde_VAG1e(SGn)
s = SGn is een wachtstandrichting
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar
u = WG(SGn)
v = AFK(SGn)
ba = MVG(SGn)
sj = RVAG2e(SGn)
/* let op: op basis van MVG(SGn) kunnen conflicterende */
/* signaalgroepen rood-voor-groen zijn geworden */

⇒ **van VAG2e naar WG**

reset_VAG2e(SGn)
en set_WG(SGn) als j && bj waar is

j = VAG2e(SGn)
bj= RWG_OP(SGn)

⇒ **van VAG2e naar MVG**

reset_VAG2e(SGn)
en set_MVG(SGn) als w && (x || y || q || ah && ak)
waar is

w = VAG2e(SGn) && !VVAG2e(SGn)
x = er is geen detector aan SGn toegewezen met een H2e_functie of er is geen hiaattijd voor detectoren met een H2e_functie die aan SGn zijn toegewezen gedefinieerd
y = !H2e_loopt(SGn) || AFK(SGn)
In de bovenstaande formule staat H2e_loopt(SGn) voor het lopen van een hiaattijd van een detector met een H2e_toewijzing die is toegewezen aan SGn.
q = !MG_TIJD(SGn)
ah = VNM(SGn)
ak = voor een met SGn conflicterende SGj is RVG(SGj) waar

⇒ **van MVG naar WG**

reset_MVG(SGn)
en set_WG(SGn) als ba && bj waar is

ba= MVG(SGn)
bj= RWG_OP(SGn)

⇒ **einde MVG**

reset_MVG(SGn)

als z && aa && (!ab || ac && ad || ax && ay && az || v) || ba && sj
waar is

z = MVG(SGn) && !VMVG(SGn)
aa = !GG_TIJD(SGn)
ab = TMVG(SGn)
ac = voor een met SGn conflicterende richting j of een SGj zoals in de applicatiespecificatie is opgegeven is RVG(SGj) waar
ad: indien !BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is WR(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
indien BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is R(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
ax = er zijn wachtstandrichtingen gedefinieerd
ay = voor alle wachtstandrichtingen j is
WR(SGj) && !GR_TIJD(SGj) && !PG(SGj, BKl) && !TR(SGj) && !A(SGj) || ROG(SGj) && !A(SGj) || WG(SGj)
waar
az = voor alle niet-wachtstandrichtingen k is
MVG(SGk) && !begin_MVG(SGk) && !H1e_loopt(SGn) && !H2e_loopt(SGn) || !G(SGk) && !A(SGk)
waar
v = AFK(SGn)
ba = MVG(SGn)
sj = RVAG2e(SGn)
/* let op: op basis van MVG(SGn) kunnen conflicterende */
/* signaalgroepen rood-voor-groen zijn geworden */

⇒ **begin GL**

reset_G(SGn)
en set_GL(SGn) als z && aa && (!ab || ac && ad || ax && ay && az || v)
waar is

z = MVG(SGn) && !VMVG(SGn)
aa = !GG_TIJD(SGn)
ab = TMVG(SGn)
ac = voor een met SGn conflicterende richting j of een SGj zoals in de applicatiespecificatie is opgegeven is RVG(SGj) waar
ad: indien !BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is WR(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
indien BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is R(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
ax = er zijn wachtstandrichtingen gedefinieerd
ay = voor alle wachtstandrichtingen j is
WR(SGj) && !GR_TIJD(SGj) && !PG(SGj, BKl) && !TR(SGj) && !A(SGj) || ROG(SGj) && !A(SGj) || WG(SGj)
waar
az = voor alle niet-wachtstandrichtingen k is
MVG(SGk) && !begin_MVG(SGk) && !H1eTIJD(SGn) && !H2eTIJD(SGn) || !G(SGk) && !A(SGk)
waar
v = AFK(SGn)

⇒ **van GL naar R**

reset_GL(SGn)
en set_R(SGn)
als ae
waar is

ae = GL(SGn) && !GL_TIJD(SGn) && !GGL_TIJD(SGn)

⇒ **begin WR**

```
set_WR(SGn)        als     ae || (b || br) && (c || si)
                   waar is

        /*  zie ook blokprocedure  */

        b =    RVG(SGn)
        br =   ROG(SGn)
        c =    VNM(SGn) || BL(SGn)
        ae =   GL(SGn) && !GL_TIJD(SGn) &&  !GGL_TIJD(SGn)
        si = RWR(SGn)
                /* let op: op basis van RVG(SGn) kan meeverlenggroen */
                /*          van andere signaalgroepen zijn beëindigd       */
```

5.3 Overgangsformules in regeling zonder blokkenschema

De hierna gebruikte afkortingen a, b, c, d, ... enz. gelden slechts binnen hoofdstuk 5.

⇒ **van WR naar ROG**

reset_WR(SGn)
en set_ROG(SGn) als se && sf
waar is

se = WR(SGn) && !GR_TIJD(SGn) && !TR(SGn)
sf = BEURT(SGn)

⇒ **BEURT**

set_BEURT als sa && sb
waar is

sa = de instructie act_BEURT(SGn) wordt gegeven
sb = WR(SGn)

reset_BEURT als sc || sd
waar is

sc = de overgang van ROG(SGn) naar RVG(SGn) wordt gemaakt
sd = de overgang van ROG(SGn) naar WR(SGn) wordt gemaakt

⇒ **einde ROG**

reset_ROG(SGn) als a || sg && (sh || si)
waar is

a = ROG(SGn) && A(SGn)
sg = ROG(SGn) && !begin_ROG(SGn) && !A(SGn)
sh = !ROG_OP(SGn)
si = RWR_OP(SGn)
NB: de set en de reset van de RWR_OP() moet geregeld worden in de applicatie-specificatie

⇒ **begin RVG**

set_RVG(SGn) als a
waar is

a = ROG(SGn) && A(SGn)

⇒ **einde RVG**

reset_RVG(SGn) als b && (c || !c && d && e && av && !aw)
waar is

⇒ **einde R, begin G en begin VG**

reset_R(SGn)
set_G(SGn)
en set_VG(SGn) als b && !c && d && e && av && !aw
waar is

b = RVG(SGn)
c = BL(SGn)
d = voor alle in de conflictmatrix van de applicatiespecificatie gedefinieerde conflictrichtingen j van SGn is R(SGj) waar
e = voor alle ontruimingstijden naar SGn is !O_TIJD(SGj, SGn) waar
av = voor alle garantie-ontruimingstijden naar SGn is !GO_TIJD(SGj, SGn) waar
aw = RVG_OP(SGn)

⇒ **van VG naar VAG1e**

reset_VG(SGn)
en set_VAG1e(SGn)
als af && ag
waar is

af = VG(SGn)
ag = !VG_TIJD(SGn) || AFK(SGn)

⇒ **einde VAG1e**

reset_VAG1e(SGn) als h && (o || p || q)
waar is

h = VAG1e(SGn)
o = de H1e_TIJD(SGn) is niet gedefinieerd
p = !H1e_TIJD(SGn) && !D(SGn_i) || AFK(SGn)
in de bovenstaande formule staat SGn_i voor de detectoren die daartoe in de applicatiespecificatie zijn gedefinieerd
q = !MG_TIJD(SGn)

⇒ **begin WG**

set_WG(SGn) als r && s && t
waar is

r = einde_VAG1e(SGn)
s = SGn is een wachtstandrichting
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar

⇒ **einde WG**

reset_WG(SGn) als u && (!t || v)
waar is

u = WG(SGn)
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar
v = AFK(SGn)

⇒ **begin VAG2e**

set_VAG2e(SGn) als r && (!s || !t) || u && (!t || v) || ba && sj
waar is

r = einde_VAG1e(SGn)
s = SGn is een wachtstandrichting
t = voor alle conflictrichtingen k van SGn en alle SG'en k zoals in de applicatiespecificatie zijn opgegeven is !A(SGk) waar
u = WG(SGn)
v = AFK(SGn)
ba = MVG(SGn)
sj = RVAG_OP(SGn)
NB: de set van de RWR_OP() moet geregeld worden in de applicatie-specificatie

⇒ **reset RVAG optie**

reset_RVAG_OP(SGn)
als ba && sj
waar is

ba = MVG(SGn)
sj = RVAG_OP(SGn)
NB: de set van de RWR_OP() moet geregeld worden in de applicatie-specificatie

⇒ **van VAG2e naar MVG**

reset_VAG2e(SGn)
en set_MVG(SGn) als w && (x || y || q)
waar is

w = VAG2e(SGn) && !VAG_OP(SGn)
x = de H2e_TIJD(SGn) is niet gedefinieerd
y = !H2e_TIJD(SGn) || AFK(SGn)
q = !MG_TIJD(SGn)

⇒ **einde MVG**

reset_MVG(SGn)

als z && aa && (!ab || ac && ad || ax && ay && az || v)
|| ba && sj
waar is

z = MVG(SGn) && !VMG_OP(SGn)

aa = !GG_TIJD(SGn)

ab = MVG_OP(SGn)

ac = voor een met SGn conflicterende richting j of een SGj zoals in de applicatiespecificatie is opgegeven is RVG(SGj) waar

ad: indien !BR(SGj) waar is geldt

ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is WR(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar

indien BR(SGj) waar is geldt

ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is R(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar

ax = er zijn wachtstandrichtingen gedefinieerd

ay = voor alle wachtstandrichtingen j is
WR(SGj) && !GR_TIJD(SGj) && !PG(SGj, BKl) && !TR(SGj) &&
!A(SGj) || ROG(SGj) && !A(SGj) || WG(SGj)
waar

az = voor alle niet-wachtstandrichtingen k is
MVG(SGk) && !begin_MVG(SGk) && !H1eTIJD(SGn) &&
!H2eTIJD(SGn) || !G(SGk) && !A(SGk)
waar

v = AFK(SGn)

ba = MVG(SGn)

sj = RVAG_OP(SGn)

NB: de set van de RWR_OP() moet geregeld worden in de applicatie-specificatie

⇒ **begin GL**

reset_G(SGn)
en set_GL(SGn) als z && aa && (!ab || ac && ad || ax && ay && az || v)
waar is

z = MVG(SGn) && !VMG_OP(SGn)
aa = !GG_TIJD(SGn)
ab = MVG_OP(SGn)
ac = voor een met SGn conflicterende richting j of een SGj zoals in de applicatiespecificatie is opgegeven is RVG(SGj) waar
ad: indien !BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is WR(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
indien BR(SGj) waar is geldt
ad = voor alle conflictrichtingen k van de onder ac genoemde SGj is R(SGk) || MVG(SGk) && !GG_TIJD(SGk) || GL(SGk) waar
ax = er zijn wachtstandrichtingen gedefinieerd
ay = voor alle wachtstandrichtingen j is
WR(SGj) && !GR_TIJD(SGj) && !PG(SGj, BKl) && !TR(SGj) && !A(SGj) || ROG(SGj) && !A(SGj) || WG(SGj)
waar
az = voor alle niet-wachtstandrichtingen k is
MVG(SGk) && !begin_MVG(SGk) && !H1eTIJD(SGn) && !H2eTIJD(SGn) || !G(SGk) && !A(SGk)
waar
v = AFK(SGn)

⇒ **van GL naar R**

reset_GL(SGn)
en set_R(SGn)
als ae
waar is

ae = GL(SGn) && !GL_TIJD(SGn) && !GGL_TIJD(SGn)

⇒ **begin WR**

set_WR(SGn) als ae || b && c || sg && sh || si)
waar is

b = RVG(SGn)
c = BL(SGn)
ae = GL(SGn) && !GL_TIJD(SGn) && !GGL_TIJD(SGn)
sg = ROG(SGn) && !begin_ROG(SGn) && !A(SGn)
sh = !ROG_OP(SGn)
si = RWR_OP(SGn)
NB: de set en de reset van de RWR_OP() moet geregeld worden in de applicatie-specificatie

5.4 Tijden

start_GG_TIJD(SGn) als begin_G(SGn)
waar is

herstart_VG_TIJD(SGn) als begin_G(SGn)
waar is

herstart_MG_TIJD(SGn) als (begin_G(SGn) || begin_WG(SGn) || einde_WG(SGn))
waar is

halteer_MG_TIJD(SGn) als begin_WG(SGn)
waar is

herstart_B_TIJD(SGn_i) als begin_D(SGn_i)
waar is

herstart_H_TIJD(SGn_i) als (begin_D(SGn_i) || einde_D(SGn_i))
waar is (als betreffende detector een H1e_spec_toewijzing of H2e_spec_toewijzing heeft, wordt de H_TIJD niet geherstart)

halteer_H_TIJD(SGn_i) als begin_D(SGn_i)
waar is (als betreffende detector een H1e_spec_toewijzing of H2e_spec_toewijzing heeft, wordt de H_TIJD niet gehalteerd)

in de hierbovenstaande instructies staat SGn_i voor de detectoren die bij de betreffende hiaattijd in de applicatiespecificatie zijn gedefinieerd

start_GL_TIJD(SGn)	als begin_GL(SGn) waar is
start_GGL_TIJD(SGn)	als begin_GL(SGn) waar is
start_GR_TIJD(SGn)	als begin_R(SGn) waar is
start_O_TIJD(SGn, SGj)	als begin_R(SGn) waar is
start_GO_TIJD(SGn, SGj)	als begin_R(SGn) waar is

5.4 Diversen

⇒ **set en reset A**

set_A(SGn) als (DA(SGn) || MA(SGn) || BA(SGn)) && !einde_BLOK(BKl) && !BL(SGn) && !VNM(SGn)
waar is
(de onderstreepte termen gelden niet in een regeling zonder blokkenschema)

reset_A(SGn) als A(SGn) && (begin_G(SGn) || !ROG(SGn) && (VNM(SGn) || BL(SGn)))
waar is
(de onderstreepte termen gelden niet in een regeling zonder blokkenschema)

⇒ **set en reset BA**

set_BA(SGn) als bb && ap && bm || !bb && ap && aq && bm
waar is

bb = SGn is een voetgangersrichting of (brom)fietsrichting
ap = R(SGn)
bm = voor een detector met een aanvraagfunctie toegewezen aan SGn waarvoor een bezettijd is opgegeven geldt: D(SGn_i) && !B_TIJD(SGn_i)
aq = !GR_TIJD(SGn)

reset_BA(SGn) als BA(SGn) && begin_G(SGn)
waar is

⇒ **set en reset DA**

set_DA(SGn) als bb && ap && (ao || at) || !bb && ap && (ao || at && aq) || bt
waar is

bb = SGn is een voetgangersrichting of (brom)fietsrichting
ap = R(SGn)
ao = de drukknop(pen) met een aanvraagvariabele voor SGn waarvoor geen bezettijd is opgegeven geeft (geven) een melding
at = de detector(en) met een aanvraagvariabele voor SGn waarvoor geen bezettijd is opgegeven geeft (geven) een melding
aq = !GR_TIJD(SGn)
bt = voor SGn geldt BA(SGn)

reset_DA(SGn) als DA(SGn) && begin_G(SGn)
waar is

⇒ **set en reset MA**

set_MA(SGn) als ar || bc && aq && as
waar is

ar = MA_OP(SGn) && (WR(SGn) || ROG(SGn)) && !VNM(SGn) && !BL(SGn)
bc = SGn is een wachtstandrichting en voor SGn is ROG(SGn) waar
as = voor alle met de wachtstandrichtingen conflicterende richtingen j is R(SGj) && !GR_TIJD(SGj) && !einde_GR_TIJD(SGj) && !A(SGj) && ba waar
ba = voor alle ontruimingstijden van de onder as genoemde SG'en j is !O_TIJD(SGj, SGn) waar

(de onderstreepte termen gelden niet in een regeling zonder blokkenschema)

reset_MA(SGn) als MA(SGn) && (begin_G(SGn) || VNM(SGn) || BL(SGn))
waar is

(de onderstreepte termen gelden niet in een regeling zonder blokkenschema)

⇒ **reset RWG_OP**

reset_RWG_OP(SGn) als WG(SGn) && !begin_WG(SGn) && RWG_OP(SGn)

⇒ **enkele opties**

AFK(SGn) = AFK_OP(SGn) && !(ROG(SGn) && A(SGn) || RVG(SGn) || GG_TIJD(SGn))

BL(SGn) = BL_OP(SGn)

TMVG(SGn) = TMVG_OP(SGn)

PPB(SGn) = PPB_OP(SGn)

PPS(SGn) = PPS_OP(SGn)

PPV(SGn) = PPV_OP(SGn)

RVAG2e(SGn) = RVAG2e_OP(SGn)

VRVG(SGn) = VRVG_OP(SGn)

set_RWR(SGn) = RWR_OP(SGn) && (ROG(SGn) || RVG(SGn))
reset_RWR(SGn) = begin_WR(SGn)

TR(SGn) = TR_OP(SGn)

VAA(SGn) = VAA_OP(SGn)

VVAG2e(SGn) = VVAG2e_OP(SGn)

VMVG(SGn) = VMVG_OP(SGn)

HOOFDSTUK 6 Blokprocedure

6.1 Algemeen

a) Bij het inschakelen van het regelprogramma worden alle toestanden en variabelen in de stand "niet waar" gezet.
Alle tijden worden in de stand "niet lopen" gezet.
Alle tellers worden op “0” gezet.

b) Na de alles roodtijd na inschakelen
b.1) worden van alle signaalgroepen de toestanden
WR(SGn) en R(SGn) waar gemaakt en
b.2) wordt BLOK(BK1) waar gemaakt.

6.2 Blokwisseling

De hierna gebruikte afkortingen a, b, c, d, ... enz. gelden slechts binnen hoofdstuk 6.

⇒ **blokophoud-optie**

BO(BKl) = BO_OP(BKl)

⇒ **set en reset PPP**

set_PPP(SGn, BKm)
als ROG(SGn) && PPP_OP(SGn, BKm) && BLOK(BKm)
waar is

reset_PPP(SGn, BKm)
als !PPP_OP(SGn, BKm)
waar is

⇒ **set en reset PPA**

set_PPA(SGn, BKm)
als c && f && !g
waar is

c = BLOK(BKm)
f = PPA_OP(SGn, BKm)
g = voor alle primaire SG'en k van blok m is
PG(SGk, BKm) && (WR(SGk) || ROG(SGk) && !A(SGk) &&
!PPP(SGk, BKm) || WG(SGk) || MVG(SGk) || GL(SGk)) waar

reset_PPA(SGn, BKm)
als !f || c && g
waar is

c = BLOK(BKm)
f = PPA_OP(SGn, BKm)
g = voor alle primaire SG'en k van blok m is
PG(SGk, BKm) && (WR(SGk) || ROG(SGk) && !A(SGk) &&
!PPP(SGk, BKm) || WG(SGk) || MVG(SGk) || GL(SGk)) waar

⇒ **set en reset BLOK**

reset_BLOK(BKm) als dj && d && e && !ca && !cf ||
dj && d && e && ca && !cf && (cb || cc || cd)
waar is

dj = BLOK(BKm) && !begin_BLOK(BKm)
d = voor alle primaire SG'en k van blok m is
PG(SGk, BKm) && (WR(SGk) || ROG(SGk) && !A(SGk) &&
!PPP(SGk, BKm) || RVG(SGk) || G(SGk) || GL(SGk)) waar
e = voor alle gespecificeerde alternatieve SG'en j van blok m is !PPA(SGj, BKm)
waar
ca = het in het blokkenschema gedefinieerde wachtblok (WB) is actief
cb = voor een niet in het wachtblok gedefinieerde niet-wachtstandrichting of voor een alternatief in het wachtblok gedefinieerde niet wachtstandrichting is A(SGn)
waar
cc = voor een niet-wachtstandrichting is
WR(SGn) && PG(SGk, BKl) waar
cd = voor een wachtstandrichting is
WR(SGn) || GL(SGn) waar
als er geen wachtstandrichtingen zijn gedefinieerd is deze formule per definitie niet waar
ce = BO (BKm)
gg = voor alle primaire SG'en k van blok m is
PG(SGk, BKm) && (WR(SGk) || ROG(SGk) && A(SGk) || RVG(SGk) || G(SGk) || GL(SGk)) waar

set en reset cf:
set_cf als c && ce && !gg
waar is
reset_cf als !ce || c && gg
waar is

set_BLOK(BKm) als einde_BLOK(BKl)
waar is

in deze formule is l het nummer van het blok dat voorafgaat aan blok m in het blokken-schema
/* voor PG(SGn, BKl) en voor PR(SGn, BKl) */
/* zie 6.3 en verder */

6.3 Primaire signaalgroepen van het actieve blok

6.3.1 Realisatie

⇒ **van WR naar ROG**

reset_WR(SGn)
set_ROG(SGn)
set_PG(SGn, BKm)
en set_PR(SGn, BKm)
als h && c && o && p
waar is

h = SGn is een primaire SG van blok m
c = BLOK(BKm)
o = WR(SGn) && !PG(SGn, BKm) && !GR_TIJD(SGn) && !TR(SGn)
p = voor alle conflictrichtingen k van SGn is
WR(SGk) || WG(SGk) || MVG(SGk) || GL(SGk) || VNM(SGk) waar

/* zie ook 6.6 */

⇒ **van ROG naar WR**

reset_ROG(SGn)
en set_WR(SGn) als h && (q || sg && si) && r
waar is

h = SGn is een primaire SG van blok m
q = einde_BLOK(BKm)
r = ROG(SGn)
sg = !begin_ROG(SGn) && !A(SGn)
si = RWR(SGn)

⇒ **reset PG**

reset_PG(SGn, BKm)
als h && q
waar is

h = SGn is een primaire SG van blok m
q = einde_BLOK(BKm)

⇒ **reset PR**

reset_PR(SGn, BKl)
als begin_WR(SGn) && WR(SGn)
waar is

6.3.2 Overnames

⇒ **van primair in blok l naar primair in blok m**

reset_PR(SGn, BKl)
set_PR(SGn, BKm)
en set_PG(SGn, BKm)
als h && s && xp
waar is

h = SGn is een primaire SG van blok m
s = PR(SGn, BKl) && !BLOK(BKl) && BLOK(BKm) && (ROG(SGn) || RVG(SGn) || G(SGn))
xp = (voor alle conflictrichtingen j van SGn is
WR(SGj) || MVG(SGj) || GL(SGj) waar) &&
!TO_PRIM(SGn)

/* zie ook 6.6 */

⇒ **van primair in blok m (oude cyclus) naar primair in blok m (nieuwe cyclus)**

set_PG(SGn, BKm)
als c && h && hh && xp
waar is

c = BLOK(BKm)
h = SGn is een primaire SG van blok m
hh = !PG(SGn, BKm) && PR(SGn, BKm) && (ROG(SGn) || RVG(SGn) || G(SGn))
xp = (voor alle conflictrichtingen j van SGn is
WR(SGj) || MVG(SGj) || GL(SGj) waar) &&
!TO_PRIM(SGn)

/* zie ook 6.6 */

6.4 Primaire signaalgroepen van een niet actief blok

6.4.1 Realisatie

⇒ **van WR naar ROG**

reset_WR(SGn)
set_ROG(SGn)
set_PG(SGn, BKm⊕b)
en set_PR(SGn, BKm⊕b)

als t && u && v && (w && ag || ww) && (x && ag || xx) && y && z
waar is

t = SGn is een primaire SG van blok m⊕b

u = WR(SGn) && A(SGn) && !PG(SGn, BKm⊕b) && !GR_TIJD(SGn) && !BLOK(BKm⊕b) && !TR(SGn)

v = voor alle conflictrichtingen j van SGn is
WR(SGj) || ROG(SGj) && !A(SGj) || MVG(SGj) || GL(SGj) || VNM(SGj) waar

w = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !VNM(SGk) && (MVG(SGk) || GL(SGk)) waar

ag = PPV(SGn)

ww = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || MVG(SGk) || GL(SGk)) waar

x = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !WR(SGk) && !ROG(SGk) waar

xx = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !WR(SGk) && !ROG(SGk)) waar

y = voor alle alternatieve conflictrichtingen k en alle alternatieve fictieve conflictrichtingen k van SGn in het actieve blok m is
!PPA(SGk, BKm) waar

z = SGn kan niet als alternatieve SG van blok m of als primaire SG van één van de blokken m tot m⊕b de overgang van WR(SGn) naar ROG(SGn) maken
/* zie ook 6.6 */

6.4.2 Reactie van de SG'en k

⇒ **set PG**

set_PG(SGk, BKl) als aa && ab
waar is

aa = begin_PR(SGn, BKm⊕b)
/* zie 6.4.1, 6.5.3 en 6.7.2 */
ab = SGk is een bij 6.4.1, 6.5.3 en 6.7.2 onder w, x, ww en xx genoemde (fictieve) conflictrichting van de onder aa genoemde SGn en l is één van de blokken m tot m⊕b

⇒ **van ROG naar WR**

reset_ROG(SGk)
en set_WR(SGk) als aa && ab && ad
waar is

aa = begin_PR(SGn, BKm⊕b)
/* zie 6.4.1, 6.5.3 en 6.7.2 */
ab = SGk is een bij 6.4.1, 6.5.3 en 6.7.2 onder w, x, ww en xx genoemde (fictieve) conflictrichting van de onder aa genoemde SGn en l is één van de blokken m tot m⊕b
ad = ROG(SGk)

6.5 Alternatieve signaalgroepen van het actieve blok

6.5.1 Realisatie

⇒ **van WR naar ROG**

reset_WR(SGn)
set_ROG(SGn)
en set_AR(SGn, BKm)
als ah && c && ao && v && ap && aq && ar
waar is

ah = SGn is een alternatieve SG van blok m
c = BLOK(BKm)
ao = WR(SGn) && !GR_TIJD(SGn) && A(SGn) && PPA(SGn, BKm) && !TR(SGn)
v = voor alle conflictrichtingen j van SGn is
WR(SGj) || ROG(SGj) && !A(SGj) || MVG(SGj) || GL(SGj) || VNM(SGj) waar
ap = voor alle primaire conflictrichtingen k van SGn in blok m is
WR(SGk) && (PG(SGk, BKm) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKm) || !VNM(SGk) && (MVG(SGk) ||
GL(SGk)) waar
aq = voor alle primaire fictieve conflictrichtingen k van SGn in blok m is
WR(SGk) && (PG(SGk, BKm) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKm) || !WR(SGk) && !ROG(SGk) waar
ar = voor de eventueel als primaire SG in blok m gespecificeerde SGn is WR(SGn)
&& PG(SGn, BKm) waar

/* zie ook 6.6 */

⇒ **reset AR**

reset_AR(SGn, BKl)
als begin_WR(SGn) && WR(SGn)
waar is

6.5.2 Reactie van de SG'en k

⇒ **set PG**

set_PG(SGk, BKm) als as && at
waar is

as = begin_AR(SGn, BKm)
/* zie 6.5.1 en 6.5.3 */
at = SGk is een bij 6.5.1 en 6.5.3 onder ap of aq genoemde (fictieve) conflictrichting van de onder as genoemde SGn

⇒ **van ROG naar WR**

reset_ROG(SGk)
en set_WR(SGk) als as && at && ad
waar is

as = begin_AR(SGn, BKm) /* zie 6.5.1 en 6.5.3 */
at = SGk is een bij 6.5.1 en 6.5.3 onder ap of aq genoemde (fictieve) conflictrichting van de onder as genoemde SGn
ad = ROG(SGk)

6.5.3 Overnames

⇒ **van alternatief in blok l naar primair in blok m**

reset_AR(SGn, BKl)
set_PR(SGn, BKm)
en set_PG(SGn, BKm)
als aw && ax && ay && h && xp
waar is

aw = AR(SGn, BKl)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
ay = !BLOK(BKl) && BLOK(BKm)
h = SGn is een primaire SG van blok m
xp = (voor alle conflictrichtingen j van SGn is
WR(SGj) || MVG(SGj) || GL(SGj) waar) &&
!TO_PRIM(SGn)
/* zie verder ook 6.3.1 en 6.6 */

⇒ **van alternatief in blok l naar alternatief in blok m**

reset_AR(SGn, BKl)
en set_AR(SGn, BKm)
als aw && ax && ay && az && ap && aq && !h
waar is

aw = AR(SGn, BKl)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
ay = !BLOK(BKl) && BLOK(BKm)
az = SGn staat in blok m als alternatieve SG gespecificeerd en PPA(SGn, BKm) is waar
ap = voor alle primaire conflictrichtingen k van SGn in blok m is
WR(SGk) && (PG(SGk, BKm) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKm) || !VNM(SGk) && (MVG(SGk) ||
GL(SGk)) waar
aq = voor alle primaire fictieve conflictrichtingen k van SGn in blok m is
WR(SGk) && (PG(SGk, BKm) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKm) || !WR(SGk) && !ROG(SGk) waar
h = SGn is een primaire SG van blok m
/* zie verder ook 6.5.1, 6.5.2 en 6.6 */

⇒ **van alternatief in blok l naar primair in blok m⊕b**

reset_AR(SGn, BKl)
set_PR(SGn, BKm⊕b)
en set_PG(SGn, BKm⊕b)
als aw && ax && bb && t && (w && ag && dh || ww) &&
(x && ag && dh || xx) && y && bc && bd
waar is

aw = AR(SGn, BKl)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
bb = !BLOK(BKm⊕b) && VNM_OP(SGn, BKl) && !TO_PRIM(SGn)
t = SGn is een primaire SG van blok m⊕b
w = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !VNM(SGk) && (MVG(SGk) ||
GL(SGk)) waar
ag = PPV(SGn)
dh = VNM_OP(SGn, BKl)
ww = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
MVG(SGk) || GL(SGk)) waar
x = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !WR(SGk) && !ROG(SGk) waar
xx = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
!WR(SGk) && !ROG(SGk)) waar
y = voor alle alternatieve conflictrichtingen k en alle alternatieve fictieve
conflictrichtingen k van SGn in het actieve blok m is
!PPA(SGk, BKm) waar
bc = voor SGn kan niet volgens de hiervoor gespecificeerde overname's PR(SGn,
BKm) of AR(SGn, BKm) worden geset
bd = SGn staat niet primair gespecificeerd in de blokken m⊕1 tot m⊕b

/* zie verder ook 6.3.1, 6.4.2 en 6.6 */

6.5.4 Versneld naar MVG of WR

⇒ **set en reset VNM**

set_VNM(SGn) als aw && br && (bf || !ta) && bc && bg
waar is

aw = AR(SGn, BKl)
br = ROG(SGn) && A(SGn) || RVG(SGn) ||
G(SGn) && !GG_TIJD(SGn) && !MVG(SGn)
bf = VNM_OP(SGn, BKl) && BLOK(BKm)
bc = voor SGn kan niet volgens 6.5.3 PR(SGn, BKm) of AR(SGn, BKm) worden geset
bg = voor SGn kan niet volgens 6.5.3 PR(SGn, BKm⊕b) worden geset
ta = SGn is een alternatieve SG van blok l

reset_VNM(SGn) als bh || bo && bp || bq
waar is

bh = (WR(SGn) && !GR_TIJD(SGn) || einde_GR_TIJD(SGn)) && VNM(SGn)
bo = MVG(SGn)
bp = voor alle met SGn conflicterende richtingen j is
!RVG(SGj) waar
bq = G(SGn) && !MVG(SGn) && VNM(SGn) && einde_AR(SGn, BKl)

6.6 Gekoppelde signaalgroepen

De overgang van WR(SGn) naar ROG(SGn) (de realisatie van een SG) kan in de applicatie-specificatie worden gekoppeld aan de gelijksoortige overgang van andere signaalgroepen.
Indien deze voorwaarde is gespecificeerd mag de overgang van WR(SGn) naar ROG(SGn) uitsluitend plaatsvinden, indien gelijktijdig de gelijksoortige overgang in dezelfde realisatievorm voor de gespecificeerde andere SG'en kan plaatsvinden.
Hetzelfde geldt in omgekeerde richting voor de overgang van WR(SGj) naar ROG(SGj); hierin staat SGj voor de gespecificeerde andere SG'en.
Het bovenstaande geldt voor de volgende realisatievormen:
- de realisatie als primaire SG van het actieve blok,
- de realisatie als alternatieve SG van het actieve blok en
- de realisatie als primaire SG van het blok m⊕b

en niet voor de bijzondere en de speciale realisatie.

De overname van een realisatievorm van SGn in een andere realisatievorm van SGn kan in de applicatiespecificatie worden gekoppeld aan de gelijksoortige overname van een realisatievorm bij andere signaalgroepen.
Indien deze voorwaarde is gespecificeerd mag een dergelijke overname uitsluitend plaatsvinden, indien gelijktijdig dezelfde overname voor de gespecificeerde andere SG'en kan plaatsvinden.
Hetzelfde geldt in omgekeerde richting voor de overname van de gespecificeerde andere SG'en.
Het bovenstaande met betrekking tot de overname van een realisatievorm geldt voor alle overnamevormen die de basisspecificatie kent.

6.7 Bijzondere realisatie buiten het blokkenschema

6.7.1 Realisatie

⇒ **van WR naar ROG**

reset_WR(SGn)
set_ROG(SGn)
en set_BR(SGn) als da && db && df
waar is

da = WR(SGn) && !GR_TIJD(SGn) && A(SGn) && PPB (SGn) && !TR(SGn)
db = SGn kan niet als primaire SG van BLOKm de overgang van WR naar ROG maken
df = voor alle conflictrichtingen j van SGn is
!GO_TIJD(SGn, SGj) && !O_TIJD(SGn, SGj) waar

⇒ **reset BR**

reset_BR(SGn) als begin_WR(SGn) && WR(SGn)
waar is

6.7.2 Overnames

⇒ **van bijzonder naar primair in blok m**

reset_BR(SGn)
set_PR(SGn, BKm)
en set_PG(SGn, BKm)
als dc && ax && c && h && xp
waar is

dc = BR(SGn)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
c = BLOK(BKm)
h = SGn is een primaire SG van blok m
xp = (voor alle conflictrichtingen j van SGn is
WR(SGj) || MVG(SGj) || GL(SGj) waar) &&
!TO_PRIM(SGn)

⇒ **van bijzonder naar primair in blok m⊕b**

reset BR(SGn)
set_PR(SGn, BKm⊕b)
en set_PG(SGn, BKm⊕b)

als dc && ax && dd && t && v && (w && ag || ww) && (x && ag || xx) && y && de && bd
waar is

dc = BR(SGn)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
dd = !BLOK(BKm⊕b) && !TO_PRIM(SGn)
t = SGn is een primaire SG van blok m⊕b
v = voor alle conflictrichtingen j van SGn is
WR(SGj) || ROG(SGj) && !A(SGj) || MVG(SGj) || GL(SGj) || VNM(SGj) waar
w = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !VNM(SGk) && (MVG(SGk) ||
GL(SGk)) waar
ag = PPV(SGn)
ww = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
MVG(SGk) || GL(SGk)) waar
x = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !WR(SGk) && !ROG(SGk) waar
xx = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
!WR(SGk) && !ROG(SGk)) waar
y = voor alle alternatieve conflictrichtingen k en alle alternatieve fictieve
conflictrichtingen k van SGn in het actieve blok m is
!PPA(SGk, BKm) waar
de = voor SGn kan niet volgens de hiervoor gespecificeerde overname PR(SGn,
BKm) worden geset
bd = SGn staat niet primair gespecificeerd in de blokken m tot m⊕b

/* zie verder ook 6.3.1 en 6.4.2 */

6.8 Speciale realisaties buiten het blokkenschema om

Speciale realisaties en bijzondere realisaties buiten het blokkenschema om zijn afwijkend voor wat betreft:

- voor een speciale realisatie is geen A(SGn) nodig;
- bij een speciale realisatie wordt niet gekeken naar lopende ontruimingstijden van SGn;
- met een speciale realisatie wordt bij het einde_MVG geen rekening gehouden.

Speciale realisaties zijn in principe bedoeld voor het regelen zonder blokkenschema terwijl bijzondere realisaties bedoeld zijn voor verkeersregelingen met een blokkenschema. Overigens kunnen beide realisatievormen zowel worden gebruikt in regelingen met of zonder blokkenschema.

6.8.1 Realisatie

⇒ **van WR naar ROG**

```
reset_WR(SGn)
set_ROG(SGn)
en set_SR(SGn)      als     tb && db
                    waar is
          tb =   WR(SGn) && !GR_TIJD(SGn) && PPS(SGn) && !TR(SGn)
                    /* let op: een speciale realisatie kan dus zonder aanvraag tot stand  */
                    /*          komen, dit in tegenstelling tot een alternatieve, versnelde */
                    /*          of bijzondere realisatie                                    */
          db =  SGn kan niet als primaire SG van BLOKm de overgang van WR naar ROG
maken
```

⇒ **van ROG naar WR**

```
reset_ROG(SGn)
en set_WR(SGn)      als     sg && (sh || si) && tc && r
                    waar is
        sg =   !begin_ROG(SGn) && !A(SGn)
        sh =   !PPS(SGn)
        si =    RWR(SGn)
        tc =    SR(SGn)
        r =     ROG(SGn)
/* let op: RWR(SGn) heeft dus een hogere prioriteit dan PPS(SGn) */
```

⇒ **reset SR**

reset_SR(SGn) als begin_WR(SGn) && WR(SGn)
waar is

6.8.2 Overnames

⇒ **van speciaal naar primair in blok m**

reset_SR(SGn)
set_PR(SGn, BKm)
en set_PG(SGn, BKm)
als tc && ax && c && h && xp
waar is

tc = SR(SGn)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
c = BLOK(BKm)
h = SGn is een primaire SG van blok m
xp = (voor alle conflictrichtingen j van SGn is
WR(SGj) || MVG(SGj) || GL(SGj) waar) &&
!TO_PRIM(SGn)

⇒ **van speciaal naar primair in blok m⊕b**

reset SR(SGn)
set_PR(SGn, BKm⊕b)
en set_PG(SGn, BKm⊕b)

als tc && ax && dd && t && v && (w && ag || ww) && (x && ag || xx) && y && de && bd
waar is

tc = SR(SGn)
ax = ROG(SGn) && A(SGn) || RVG(SGn) || G(SGn)
dd = !BLOK(BKm⊕b) && !TO_PRIM(SGn)
t = SGn is een primaire SG van blok m⊕b
v = voor alle conflictrichtingen j van SGn is
WR(SGj) || ROG(SGj) && !A(SGj) || MVG(SGj) || GL(SGj) || VNM(SGj) waar
w = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !VNM(SGk) && (MVG(SGk) ||
GL(SGk)) waar
ag = PPV(SGn)
ww = voor alle primaire conflictrichtingen k van SGn in de blokken l (m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
MVG(SGk) || GL(SGk)) waar
x = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
WR(SGk) && (PG(SGk, BKl) || !A(SGk) && VAA(SGk) && !VNM(SGk)) ||
ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) || !WR(SGk) && !ROG(SGk) waar
xx = voor alle primaire fictieve conflictrichtingen k van SGn in de blokken l
(m<=l<m⊕b) is
PG(SGk, BKl) && (WR(SGk) || ROG(SGk) && !A(SGk) && !PPP(SGk, BKl) ||
!WR(SGk) && !ROG(SGk)) waar
y = voor alle alternatieve conflictrichtingen k en alle alternatieve fictieve
conflictrichtingen k van SGn in het actieve blok m is
!PPA(SGk, BKm) waar
de = voor SGn kan niet volgens de hiervoor gespecificeerde overname PR(SGn,
BKm) worden geset
bd = SGn staat niet primair gespecificeerd in de blokken m tot m⊕b

/* zie verder ook 6.3.1 en 6.4.2 */

HOOFDSTUK 7 Overige specificaties

⇒ Datum, dag en tijd, dag- maand- en jaarnummer

In de applicatiespecificatie kan m.b.v. de referentie datum(x, y, z) nagegaan worden of een bepaalde datum waar is.
Als
- x overeenkomt met het werkelijke jaartal zoals in de procesbesturing aanwezig en
- y overeenkomt met de werkelijke maand zoals in de procesbesturing aanwezig en
- z overeenkomt met de werkelijke dag zoals in de procesbesturing aanwezig

dan geeft de referentie datum(x, y, z) het resultaat waar, anders niet waar.

In de applicatiespecificatie kan m.b.v. de referentie dag(x) nagegaan worden of een bepaalde dag van de week (maandag dinsdag enz.) waar is.
Als
- x overeenkomt met de werkelijke weekdag (maandag dinsdag enz.) zoals in de procesbesturing aanwezig

dan geeft de referentie dag(x) het resultaat waar, anders niet waar.

In de applicatiespecificatie kan m.b.v. de referentie tijd(x, y) nagegaan worden of een bepaald tijdstip waar is.
Als
- x overeenkomt met het werkelijke uur zoals in de procesbesturing aanwezig en
- y overeenkomt met de werkelijke minuut zoals in de procesbesturing aanwezig

dan geeft de referentie tijd(x, y) het resultaat waar, anders niet waar.

De referentie dagnummer() geeft de waarde van de kalenderdag terug.
De referentie maandnummer() geeft de waarde van de kalendermaand (januari is 1, februari is 2 enz.) terug.
De referentie jaarnummer() geeft de waarde van het kalenderjaar terug.

⇒ FB

De instructie set_FB() voert een fasebewaking uit.

⇒ **FIX**

Zolang de fixatiedrukknop is ingedrukt is FIX waar, anders niet waar

⇒ **Kiesinstructies voor hiaattijden**

In de applicatiespecificatie kan voor de in te stellen waarde van een hiaattijd van een detector worden gekozen uit twee waarden per hiaattijd.
Dit is

- de waarde als vermeld in de detectietabel in de kolom ‘hiaattijd’ (H_TIJD)
- de waarde als vermeld in de detectietabel in de kolom ‘extra_hiaattijd’ (E_H_TIJD).

De vigerende ingestelde waarde wordt gekozen m.b.v. de volgende instructies.

kies_H_TIJD(SGn_i)	vanaf het moment van waar zijn van deze instructie geldt voor SGn_i als ingestelde waarde voor de hiaattijd de waarde zoals vermeld in de detectietabel in de kolom ‘hiaattijd’ (H_TIJD)
kies_E_H_TIJD(SGn)	vanaf het moment van waar zijn van deze instructie geldt voor SGn als ingestelde waarde voor de hiaattijd de waarde zoals vermeld in de detectietabel in de kolom ‘extra_hiaattijd’ (E_H_TIJD)

Default worden bij initialisatie de waarden gekozen als vermeld in de detectietabel in de kolom ‘hiaattijd (H_TIJD).

⇒ **Kiesinstructies voor maximumgroentijden**

In de appliatiespecificatie kan voor de in te stellen waarden van de maximumgroentijden worden gekozen uit maximaal zes waarden per signaalgroep.
Dit zijn

- de waarden als vermeld bij de MG_TIJD òf
- de waarden als vermeld bij de MG_TIJD1 t/m MG_TIJD12.

De keuze vindt tegelijkertijd plaats voor alle signaalgroepen.
De vigerende ingestelde waarden worden gekozen m.b.v. de volgende instructie.

kies_MG_TIJDj()	vanaf het moment van waar zijn van deze instructie geldt voor alle SG'en als ingestelde waarde voor de maximumgroentijd de waarde zoals vermeld bij de MG_TIJDj.

Default worden voor de maximumgroentijd van een signaalgroep bij initialisatie de waarde gekozen zoals vermeld bij de MG_TIJD. Indien bij de MG_TIJD NG is ingevuld, wordt default de waarde gekozen als vermeld bij de MG_TIJD1. Indien ook bij MG_TIJD1 NG is ingevuld wordt default de waarde 0 ingevuld.

⇒ **PPAH**

```
PPAH(SGn) = A(SGn) && (WR(SGn) && !PG(SGn, BKl) || ROG(SGn)) || RVG(SGn) ||
            G(SGn) && !MVG(SGn) && TIJD(SGn_i)

    herstart_TIJD(SGn_i)
                        als begin_G(SGn)
                        waar is
```

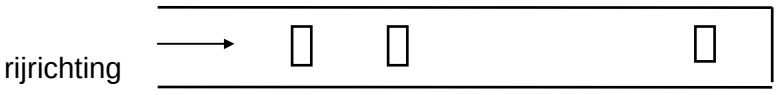
⇒ **RGAH**

```
RGAH(SGn,i)  =  MAH6(SGn,i) || MAH7(SGn,i)

    MAH4(SGn,i)  =  bb && ap || !bb && ap && aq
       bb =  SGn is een voetgangersrichting of (brom)fietsrichting
       ap =  R(SGn)
       aq =  !GR_TIJD(SGn)

    set_MAH5(SGn,i)       als    begin_D((SGn_j) && !D(SGn_i)
                          waar is
    reset_MAH5(SGn,i)     als    MAH5(SGn,i) && (begin_MAH6(SGn,i) || einde_D(SGn_j))
                          waar is

    set_MAH6(SGn,i)       als    MAH4(SGn,i) && begin_D(SGn_i) && MAH5(SGn,i)
                          waar is
    reset_MAH6(SGn,i)     als    MAH6(SGn,i) && begin_G(SGn)
                          waar is

    set_MAH7(SGn,i)       als    MAH4(SGn,i) && D(SGn_1) && !B_TIJD(SGn_i)
                          waar is

    reset_MAH7(SGn,i)     als    begin_G(SGn) && MAH7(SGn,i)
                          waar is

    herstart_TIJD(SGn_k)  als    begin_D(SGn_1)
                          waar is

/*  de hiervoorstaande MAH4(SGn,i) t/m MAH7(SGn,i) en
    RGAH(SGn,i) zijn bedoeld voor richtinggevoelige-aanvraag  */
```

In de applicatiespecificatie wordt voor richtinggevoelige aanvraag een detectorpaar opgegeven. Bij de hiervoorstaande detectoren staat

- SGn_i voor de als eerste genoemde detector van het desbetreffende paar en
- SGn_j voor de als tweede genoemde detector van het desbetreffende paar.

Voorbeeld van detectieconfiguratie richtinggevoelige aanvraag:



detector : SGn_j SGn_i SGn_1

wordt als volgt in de tabel voor signaalgroepen met richtinggevoelige aanvraag ingevuld:
1e detector : SGn_i
2e detector : SGn_j
koplus: SGn_1
signaalgroep: SGn

⇒ **H1e_loopt(SGn)**

H1e_loopt(SGn) is waar als de hiaattijd van een detector met een H1e_toewijzing, toegewezen aan SGn, loopt.

⇒ **H2e_loopt(SGn)**

H2e_loopt(SGn) is waar als de hiaattijd van een detector met een H2e_toewijzing, toegewezen aan SGn, loopt.

⇒ **Signalen in de interface**

Extra informatie kan m.b.v. de volgende instructie en referenties aan de procesbesturing worden aangeboden resp. van de procesbesturing worden ontvangen.

plaats_in_INT(x, a) plaatst een waarde t.b.v de procesbesturing in de interface,
x is positie in de interface
- x wordt aangegeven als: eerste_plaats_positie+i
- voor i geldt 0<=i<=62

voor de te plaatsen waarde a geldt 0<=a<=255

haal_uit_INT(y) haalt een door de procesbesturing geschreven waarde op uit de interface,
y is positie in de interface,
- y wordt aangegeven als: eerste_ophaal_positie+i
- voor i geldt 0<=i<=62

voor de op te halen waarde b geldt 0<=b<=255

nw_info_in_INT de procesbesturing heeft nieuwe informatie in de interface geschreven

⇒ **Realisaties: toewijzingen en instructies**

Het is verstandig om in de applicatiespecificatie de hieronder genoemde toewijzingsinstructucties op een gecontroleerd moment (bijvoorbeeld tijdens alles-rood) uit te voeren. Hiertoe is de instructie ‘maak_ALLES_ROOD()’ beschikbaar. Met deze instructie worden alle signaalgroepen, met inachtneming van de garantiegroentijden, naar rood gestuurd. Er wordt geen rekening gehouden met eventuele RVAG2e, VVAG2e_ of VMVG_opties. Zodra alle signaalgroepen rood tonen, wordt de variabele ALLES_ROOD_APPLICATIE waar.

Ook is het verstandig om bij wijzigingen van het blokkenschema de fasebewaking voor de betreffende signaalgroepen te herstarten. Door de wijzigingen kan het namelijk langer dan één cyclus duren voordat de betreffende signaalgroepen opnieuw kunnen realiseren.

set_PRIM_toewijzing (SGn, BKl) vanaf het moment van waar zijn van deze instructie geldt voor SGn en de daarmee volgens BS_H66 gekoppelde signaalgroepen een primaire realisatiemogelijkheid in BKl

reset_PRIM_toewijzing (SGn, BKl) vanaf het moment van waar zijn van deze instructie vervalt voor SGn en de daarmee volgens BS_H66 gekoppelde signaalgroepen de primaire realisatiemogelijkheid in BKl

Wanneer de betreffende signaalgroep een PG(SGn, BKl) heeft, worden de volgende aanvullende acties genomen:

- altijd: reset_PG(SGn, BKl)
- als ROG(SGn): reset_ROG(SGn) en
  set_WR(SGn).

Wanneer voor de betreffende signaalgroep PR(SGn, BKl) && (RVG(SGn) || G(SGn) || GL(SGn)) geldt, worden de volgende aanvullende acties genomen:

reset_PR(SGn, BKl)
set_SR(SGn).

Als er na het intrekken van de laatste primaire toewijzing (geen blokkenschema meer) geen wachtblok actief is, wordt het wachtblok ingesteld op BK1. Zodra er weer een signaalgroep primair aan het blokkenschema wordt toegewezen, wordt een om voorgaande reden ingesteld wachtblok uitgeschakeld.

set_ALT_toewijzing (SGn, BKl) vanaf het moment van waar zijn van deze instructie geldt voor SGn en de daarmee volgens BS_H66 gekoppelde signaalgroepen een alternatieve realisatiemogelijkheid in BKl

reset_ALT_toewijzing (SGn, BKl) vanaf het moment van waar zijn van deze instructie vervalt voor SGn en de daarmee volgens BS_H66 gekoppelde signaalgroepen de alternatieve realisatiemogelijkheid in BKl

Wanneer de betreffende signaalgroep een AR(SGn, BKl) heeft, worden de volgende aanvullende acties genomen:

reset_AR(SGn, BKl)
set_SR(SGn).

reset_BLOKKENSCHEMA voor alle signaalgroepen worden voor alle blokken de instructies ‘reset_PRIM_toewijzing(SGn, BKl)’ en ‘reset_ALT_toewijzing(SGn, BKl)’ uitgevoerd

set_INITIEEL_BLOKKENSCHEMA	eerst wordt de instructie ‘reset_BLOKKENSCHEMA’ waarna voor alle initiëel in de craptab.c opgenomen primaire en alternatieve realisatiemogelijkheden de instructies ‘set_PRIM_toewijzing(SGn, BKl)’ en ‘set_ALT_toewijzing(SGn, BKl)’ worden uitgevoerd LET OP: door het gebruik van deze instructie vervallen eventueel eerder doorgevoerde wijzigingen van het blokkenschema.
TO_PRIM(SGn)	niet toestaan overnames naar primair SGn

⇒ **status regelen**

De referentie status_regelen(x) geeft de status van de gekoppelde regelaar x. Als x daadwerkelijk regelt en de verbinding met x is in tact dan geeft de referentie het resultaat waar, anders niet waar.

⇒ **Toewijzingsinstructies**

Een aantal in de tabellen van de applicatiespecificatie vastgelegde toewijzingen voor signaalgroepen en detectoren kunnen worden gemanipuleerd met de volgende instructies.

set_AVR_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie geldt voor D(SGn_i) de AVR toewijzing volgens de signaalgroep-detectietabel
reset_AVR_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt voor D(SGn_i) de AVR toewijzing volgens de signaalgroep-detectietabel
set_BS_H5p_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie geldt voor D(SGn_i) de BS_H5p toewijzing volgens de signaalgroep-detectietabel
reset_BS_H5p_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt voor D(SGn_i) de BS_H5p toewijzing volgens de signaalgroep-detectietabel
set_DFA_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie geldt voor SGn de DFA toewijzing volgens de signaalgroep-detectietabel

reset_DFA_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie vervalt voor SGn de DFA toewijzing volgens de signaalgroep-detectietabel
set_DFU_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie geldt voor SGn de DFU toewijzing volgens de signaalgroep-detectietabel
reset_DFU_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie vervalt voor SGn de DFU toewijzing volgens de signaalgroep-detectietabel
set_H1e_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie geldt voor D(SGn_i) de H1e toewijzing volgens de signaalgroep-detectietabel
reset_H1e_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt voor D(SGn_i) de H1e toewijzing volgens de signaalgroep-detectietabel
set_H1e_spec_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie herstart/halteert de RWS-C-regelaar niet meer de H_TIJD van deze detector. Wel is H1e_loopt waar als de hiaattijd van de betreffende lus loopt
reset_H1e_spec_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt de H1e_spec_toewijzing van deze detector en gelden voor deze detector de toewijzingen zoals opgenomen in de detectietabel
set_H2e_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie geldt voor D(SGn_i) de H2e toewijzing volgens de signaalgroep-detectietabel
reset_H2e_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt voor D(SGn_i) de H2e toewijzing volgens de signaalgroep-detectietabel
set_H2e_spec_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie herstart/halteert de RWS-C-regelaar niet meer de H_TIJD van deze detector. Wel is H2e_loopt waar als de hiaattijd van de betreffende lus loopt
reset_H2e_spec_toewijzing_D(SGn_i)	vanaf het moment van waar zijn van deze instructie vervalt de H2e_spec_toewijzing van deze detector en gelden voor deze detector de toewijzingen zoals opgenomen in de detectietabel

set_KSH_toewijzing(HFi)	vanaf het moment van waar zijn van deze instructie geldt voor uitgaand koppelsignaal (HFi) de KSH_toewijzing volgens de koppeltabel (uitgaande koppelsignalen blijven door de procesbesturing bestuurd worden als de regelaar niet regelt)
reset_KSH_toewijzing(HFi)	vanaf het moment van waar zijn van deze instructie vervalt voor uitgaand koppelsignaal (HFi) de KSH_toewijzing volgens de koppeltabel (uitgaande koppelsignalen worden door de procesbesturing afgezet als de regelaar niet regelt)
set_KSU_toewijzing(HFi)	vanaf het moment van waar zijn van deze instructie geldt voor uitgaand koppelsignaal (HFi) de KSU_toewijzing volgens de koppeltabel (uitgaande koppelsignalen worden door de procesbesturing afgezet als de regelaar niet regelt)
reset_KSU_toewijzing(HFi)	vanaf het moment van waar zijn van deze instructie vervalt voor uitgaand koppelsignaal (HFi) de KSU_toewijzing volgens de koppeltabel (uitgaande koppelsignalen worden door de procesbesturing afgezet als de regelaar niet regelt)
set_WB_toewijzing	vanaf het moment van waar zijn van deze instructie geldt de WB toewijzing volgens de wachtbloktabel
reset_WB_toewijzing	vanaf het moment van waar zijn van deze instructie vervalt de WB toewijzing volgens de wachtbloktabel
set_WSGR_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie geldt voor SGn de WSGR toewijzing volgens de blokkentabel
reset_WSGR_toewijzing(SGn)	vanaf het moment van waar zijn van deze instructie vervalt voor SGn de WSGR toewijzing volgens de blokkentabel; voor een eventueel in WG staande richting wordt het WG beëindigd

⇒ **voer een dump uit**

De instructie voer_dump_uit() zorgt voor het aanmaken van een dump.

⇒ **VNMH**

VNMH(SGn) = WR(SGn) && PG(SGn, BKl) || MVG(SGn) || GL(SGn)

⇒ **Werkelijk regelen van de regeling**

Zolang de regeling daadwerkelijk regelt is de variabele W_REG_VRI waar, anders niet waar.

⇒ **WSRH**

WSRH = voor tenminste één SGn is
G(SGn) && !MVG(SGn) || MVG(SGn) && (H1e_loopt(SGn) || H2e_loopt(SGn)) ||
GL(SGn) || GR_TIJD(SGn) || A(SGn)
waar

/* bovenstaande WSRH is bedoeld voor meeverlengen */
/* bij wachtstandrood */

⇒ **VLOG**

In de RWS-C-regelaar is VLOG opgenomen. Om hiervan gebruik te maken moet bovenaan in het bestand crapcod.c het bestand ‘crvlog.h’ ge-include te worden:
#include "crvlog.h"

Bij de initialisatie van de regeling (init_bij_inschakelen) moet in het bestand crapcod.c de VRI-naam en de wijze van het genereren van VLOG-data opgegeven worden.

Met de instructie setVlogVRIid("naam"); wordt de naam van de VRI opgegeven.

Met de instructie setVlogMode(x); wordt de wijze van het genereren van VLOG-data opgegeven. De volgende instructies zijn hierbij mogelijk:

setVlogMode(VLOGMODE_NONE); = niet aanmaken VLOG-data
setVlogMode(VLOGMODE_REALTIME_BINAIR); = realtime stream, binair formaat
setVlogMode(VLOGMODE_FILE_BINAIR); = naar file, binair formaat
setVlogMode(VLOGMODE_REALTIME_ASCII); = realtime stream, ASCII formaat
setVlogMode(VLOGMODE_FILE_ASCII); = naar file, ASCII formaat

Documentatiepagina

Opdrachtgever(s)	RIJKSWATERSTAAT Water, verkeer en leefomgeving
Titel rapport	BASISSPECIFICATIE na integratie standaard en speciale versie verkeerstechnische specificatie van de basisregels van regelingen V1 31-03-2014Fout! Verwijzingsbron niet gevonden.
Kenmerk	DVS178/Dht/9607
Datum publicatie	31 maart 2014