movensis
ov laten werken

CRSV module

Functionele- en programmatechnische handleiding

Versie 6.01
Marcel Fick
11 oktober 2015

Inhoudsopgave

1. Inleiding

In deze handleiding wordt de regelstructuur CRSV beschreven. CRSV staat voor: “C-regelaar Speciaal met Vaste cyclustijd”. Deze regelstructuur kenmerkt zich door een vaste (netwerk)cyclustijd waarbinnen voor iedere signaalgroep een primaire realisatie gespecificeerd kan worden op basis van een stappenraster. Er kunnen meerdere stappenrasters met verschillende cyclustijden worden gespecificeerd.

De CRSV module is ontwikkeld om op relatief eenvoudige wijze “groene golven” binnen een netwerk te kunnen ontwerpen met behoud van een zo groot mogelijke voertuigafhankelijkheid.

De implementatie binnen de C-regelaar vindt plaats door de definitie van instelbare parameters in craptab.c en de aanroep van een aantal standaard functies in crapcod.c. Deze functies besturen de signaalgroep opties welke van invloed zijn op de fasecyclus afhandeling binnen de signaalgroep procedure.

Leeswijzer:
Hoofdstuk 2 beschrijft op kruispunt niveau de programma keuze en de opbouw van de stappenrasters. In hoofdstuk 3 en 4 wordt ingegaan op de afwikkeling van de primaire- en alternatieve realisaties. Hoofdstuk 5 beschrijft de individuele signaalgroep afwikkeling en hoofdstuk 6 de standaard koppelingen tussen signaalgroepen onderling. In hoofdstuk 7 worden de mogelijkheden beschreven van beïnvloeding door openbaar vervoer door middel van prioriteitsingrepen.

Hoofdstuk 8 geeft een programma technische beschrijving van de CRSV module. In hoofdstuk 9 tenslotte wordt het historische overzicht van de CRSV module gegeven.

2. Programma keuze

2.1. Informatie uitwisseling regelaars

Binnen een netwerk van geregelde kruispunten dient één regelaar aangewezen te worden als complexcoördinator. Deze regelaar bepaalt, bijvoorbeeld op basis van klokperioden, het gewenste stappenraster. Door middel van vier koppelsignalen wordt deze informatie verstuurd naar de overige regelaars. Er kunnen maximaal 15 stappenrasters worden gedefinieerd.

Indien de complexcoördinator geen geldig stappenraster selecteert wordt het default stappenraster actief. Het default stappenraster is instelbaar door middel van EPARM(DEFPRG).

Om het synchroon lopen van de verschillende regelaars te garanderen wordt er iedere cyclus op een gedefinieerd moment een 5^{e} koppelsignaal verstuurd vanuit de complexcoördinator naar de overige regelaars.

2.2. Specificatie stappenraster

De specificatie van een stappenraster binnen de CRSV module wordt vastgelegd door de volgende parameters:

- EPARM(CTYD_x) Cyclustijd stappenraster x
- EPARM(INSC_x) Inschakelpunt stappenraster x
- EPARM(OMSC_x) Omschakelpunt stappenraster x
- EPARM(SYNC_x) Synchronisatiepunt stappenraster x

De stapgrootte binnen het stappenraster is altijd 1 seconde. De actuele status van het stappenraster wordt vastgelegd in de variabele STAP. Indien er geen stappenraster actief is wordt STAP gelijk aan NUL.

Bij de start van stappenraster x wordt STAP gelijk aan EPARM(INSC_x). Iedere seconde wordt STAP met 1 opgehoogd. Indien STAP groter wordt dan EPARM(CTYD_x) dan wordt STAP gelijk aan 1.

Bij omschakeling van stappenraster x naar stappenraster y wordt het moment van omschakelen uitgesteld totdat STAP gelijk is aan EPARM(OMSC_x) tenzij EPARM(OMSC_x) gelijk is aan NUL. In dat geval wordt onmiddellijk omgeschakeld. Op het moment van omschakeling wordt STAP gelijk aan EPARM(INSC_y).

2.3. Synchronisatie

Bij de binnenkomst van het synchronisatie signaal wordt de afwijking in het stappenraster bepaald ten opzichte van de complexcoördinator. Indien het stappenraster 50% of minder “voor” loopt ten opzichte van de complexcoördinator dan zal het stappenraster gedurende maximaal 1 cyclus gaan vertragen waarbij STAP iedere 2 seconden met 1 wordt opgehoogd.

Indien het stappenraster 10% of minder “achter” loopt ten opzichte van de complexcoördinator dan zal het stappenraster gedurende maximaal 1 cyclus gaan versnellen waarbij STAP iedere 10 seconden eenmaal met 2 wordt opgehoogd.
In dit laatste geval vindt de daadwerkelijke ophoging plaats in twee opeenvolgende systeemronden, zodat referentie aan een bepaald moment in het stappenraster altijd mogelijk blijft.

In alle andere gevallen wordt STAP onmiddellijk gelijk gemaakt aan EPARM(SYNC_x).

Tijdens de omschakeling van stappenraster x naar stappenraster y wordt het synchronisatie mechanisme uitgeschakeld.

2.4. Bewaking instellingen

De instellingen van het actieve stappenraster worden als volgt bewaakt:

- Indien het omschakelpunt groter is dan de cyclustijd dan wordt het omschakelpunt gelijk aan NUL verondersteld.
- Indien het inschakelpunt groter is dan de cyclustijd dan wel gelijk aan NUL dan wordt het inschakelpunt gelijk aan 1 verondersteld.
- Indien het synchronisatiepunt groter is dan de cyclustijd dan wel gelijk aan NUL dan wordt het synchronisatie punt gelijk aan 1 verondersteld.
- Indien de instelling van het default stappenraster niet gelijk is aan een gedefinieerd stappenraster dan wordt het default stappenraster gelijk aan NUL verondersteld.
  Door stappenraster NUL te activeren wordt de CRSV module uitgeschakeld en worden er aan signaalgroepen geen realisaties meer toegekend.

2.5. Fixatie

Tijdens fixatie wordt het stappenraster gehalteerd en wordt het omschakelen van stappenraster x naar stappenraster y geblokkeerd. Het synchronisatie mechanisme wordt tijdens fixatie uitgeschakeld.

3. Primaire realisaties

3.1. Specificatie primair gebied

De specificatie van het primaire gebied van een signaalgroep binnen een stappenraster wordt vastgelegd door middel van de volgende parameters:

- EPARM(UITSTn_x) Einde uitstel SG(n) tijdens stappenraster x
- EPARM(STARTn_x) Start primair SG(n) tijdens stappenraster x
- EPARM(WACHTn_x) Start verlengen SG(n) tijdens stappenraster x
- EPARM(EINDEn_x) Einde primair SG(n) tijdens stappenraster x
- EPARM(AFKAPn_x) Einde groen SG(n) tijdens stappenraster x
- SCH(CPRIMn_x) Cyclische primaire realisatie SG(n) tijdens stappenraster x

3.1.1. Einde uitstel

Door middel van parameter EPARM(UITSTn_x) kan een moment binnen het stappenraster worden gedefinieerd waarvoor het naar groen sturen van een signaalgroep n niet mag plaatsvinden.

Om het startgroen moment samen te laten vallen met het “einde uitstel moment” wordt buiten het primaire gebied de fasecyclus overgang van WR() naar ROG() geblokkeerd tot het moment dat, rekening houdend met lopende ontruimingstijden, groenrealisatie niet meer kan plaatsvinden voor het “einde uitstel moment”.

Binnen het primaire gebied wordt de fasecyclusovergang van RVG() naar G() en alternatief realiseren geblokkeerd.

Indien EPARM(UITSTn_x) gelijk is aan NUL vervalt het uitstel.

3.1.2. Start primair

Door middel van parameter EPARM(STARTn_x) wordt de start van het primaire gebied gedefinieerd. Om het primaire startgroen moment te garanderen worden (fictieve)conflicterende signaalgroepen, afhankelijk van ingestelde ontruimingstijden etc., voor de start van het primaire gebied geblokkeerd dan wel afgebroken.
De CRSV module stelt deze zo nodig zelf de fictieve conflicten vast indien een signaalgroep onderdeel is van een gescheiden voetgangersoversteek. (zie paragraaf 6.3.)

Indien bij de start van het primaire gebied de fasecyclus toestand gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().

Indien EPARM(STARTn_x) gelijk is aan NUL vervalt de primaire realisatie.

3.1.3. Start verlengen

Door middel van parameter EPARM(WACHTn_x) kan een moment binnen het stappenraster worden gedefinieerd waarvoor de overgang van VAG2e() naar MVG() niet mag plaatsvinden. Hiertoe wordt de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd.

Indien EPARM(WACHTn_x) gelijk is aan NUL vervalt het blokkeren van de fasecyclus overgang van VAG2e() naar MVG().

3.1.4. Einde primair

Door middel van parameter EPARM(EINDEn_x) wordt het einde van het primaire gebied gedefinieerd. Het wel of niet lopen van MG_TIJD() is binnen de CRSV module niet van invloed op de fasecyclus overgang van VAG2e() naar MVG().

Verder geldt tijdens het primaire gebied dat indien het 1^e hiaat, gerekend vanaf einde_VG(), nog niet is gevallen dat zowel H1e_TIJD() als H2e_TIJD() de fasecyclus toestand VAG2e() aanhouden.

Indien EPARM(EINDEn_x) gelijk is aan NUL vervalt de primaire realisatie.

3.1.5. Einde groen

Door middel van parameter EPARM(AFKAPn_x) kan een moment binnen het stappenraster worden gedefinieerd waarna het groen van een signaalgroep dwingend, met inachtname van garantietijden, wordt beëindigd. Hiertoe wordt de fasecyclus overgang naar GL() geforceerd indien de signaalgroep groen is.

Daarnaast wordt buiten het primaire gebied de fasecyclus overgang van WR() naar ROG() geblokkeerd tenzij een gedefinieerd uitstelmoment groensturing toestaat dan wel dat groensturing zal resulteren in het versneld openen van het primaire gebied. (zie paragraaf 3.3.)

Binnen het primaire gebied wordt alternatief realiseren altijd geblokkeerd.

Indien EPARM(AFKAPn_x) gelijk is aan NUL vervalt het dwingend beëindigen van het groen.

3.1.6. Cyclische realisatie

Door middel van schakelaar SCH(CPRIMn_x) kan worden gedefinieerd dat een primaire realisatie ook plaats moet vinden indien er geen aanvraag voor groen aanwezig is.

Op het moment dat het primaire gebied opent wordt in dat geval de aanvraag zo nodig opgezet.

3.1.7. Bijzondere voorwaarden

Door middel van parameter EPARM(PROGKEUZE) kan het actieve stappenraster als volgt worden beïnvloed.

1.0 : Stappenraster LOKAAL
- Alle schakelaars SCH(CPRIMn_x) worden gelijk aan UIT verondersteld.
- Alle uitstel- en afkap momenten worden gelijk aan NUL verondersteld.
- Alle start verleng momenten worden gelijk aan NUL verondersteld.

1.1 : Stappenraster LOKAAL met versneld doorstappen
De werking is als bij LOKAAL met de volgende aanvullingen:
- Het synchronisatie mechanisme wordt uitgeschakeld.
- Een signaalgroep met een aanvraag voor groen die rood toont zal, voor zover (fictief) conflicterende signaalgroepen dit toestaan, het stappenraster versneld laten doorschakelen totdat deze richting primair groen toont.
- Een signaalgroep die groen toont buiten het primaire gebied zal, voor zover (fictief) conflicterende signaalgroepen dit toestaan, het stappenraster versneld laten doorschakelen totdat deze richting primair groen toont.

2.0 : Stappenraster CRSV (= default)

3.0 : Stappenraster STAR
- Alle schakelaars SCH(CPRIMn_x) worden gelijk aan AAN verondersteld.
- Alle start verleng momenten worden gelijk aan einde primair verondersteld.

3.2. Primair gerealiseerd

Een signaalgroep wordt als gerealiseerd beschouwd indien binnen het primaire gebied de fasecyclus overgang van VAG2e() naar MVG() wordt gemaakt of indien binnen het primaire gebied geen aanvraag voor groen aanwezig is.

3.3. Versneld openen primair gebied

Indien een signaalgroep groen is voor de start van het primaire gebied en voor alle (fictief) conflicterende signaalgroepen geldt dat groenrealisatie niet meer mogelijk is voor de start van dit primaire gebied dan wordt dit primaire gebied versneld geopend.

TTP[] wordt gelijk gemaakt aan NUL en de fasecyclus toestand wordt zo nodig “teruggezet” van MVG() naar VAG2e(). Een eventueel gedefinieerd en nog niet verstreken uitstel moment komt te vervallen.

3.4. Overgang van WR() naar ROG()

Een signaalgroep maakt de overgang van WR() naar ROG() indien het primaire gebied open is en er een aanvraag voor groen aanwezig is. In dat geval worden (fictief) conflicterende signaalgroepen waarvan de fasecyclus toestand gelijk is aan ROG() of RVG() “teruggezet” in de fasecyclus toestand WR().
Dit geldt overigens ook indien de fasecyclus toestand wordt “teruggezet” van MVG() naar VAG2e() bij de start van een primaire realisatie.

3.5. Bewaking instellingen

Indien de instelling van onderling (fictief) conflicterende signaalgroepen overlappend zijn ingesteld worden de ingestelde start primair momenten maatgevend. Dit betekent dat het groen van een primaire signaalgroep voortijdig kan worden afgebroken om het gedefinieerde startgroen moment van een (fictief) conflicterende signaalgroep te kunnen garanderen.

3.6. Bijzondere voorwaarden bij inschakelen

Bij de start van een stappenraster geldt de volgende bijzondere voorwaarde:

- Indien een signaalgroep niet groen is en er is onvoldoende primaire realisatie ruimte dan wordt deze signaalgroep onmiddellijk als gerealiseerd beschouwd.
  Er is sprake van onvoldoende ruimte indien het alternatieve maximum EPARM(ALVAn) niet gemaakt kan worden.

Deze bijzondere voorwaarde geldt ook indien als gevolg van het synchronisatie mechanisme STAP onmiddellijk gelijk wordt gemaakt aan EPARM(SYNC_x).

4. Alternatieve realisaties

4.1. Ondergrens voertuigafhankelijk groen bij alternatieve realisaties

Een signaalgroep kan alternatief realiseren indien er, als gevolg van het ontbreken van aanvragen voor (fictief) conflicterende signaalgroepen, ruimte ontstaat binnen de regeling.

Alternatieve realisaties worden alleen toegekend indien de ontstane ruimte binnen de regeling voldoende is voor het realiseren van de vastgroentijd. Voor signaalgroepen waarvoor een verlengfunctie is gedefinieerd geldt bovendien dat de ontstane ruimte binnen de regeling voldoende moet zijn voor het realiseren van de ingestelde ondergrens voertuigafhankelijk groen bij alternatieve realisatie, EPARM(ALVAn).
Aan signaalgroepen die op grond van een gedefinieerd uitstel moment (nog) niet groen mogen worden wordt nooit een alternatieve realisatie toegekend.

Indien meerdere, onderling (fictief) conflicterende, signaalgroepen alternatief kunnen realiseren worden de alternatieve realisaties toegekend op basis van het langst wachtende principe.

Een signaalgroep waaraan een alternatieve realisatie is toegekend maakt de overgang van WR() naar ROG().

4.2. Beëindigen alternatieve realisatie

De alternatieve realisatie van een signaalgroep wordt ingetrokken indien een (fictief) conflicterende primaire signaalgroep alsnog een aanvraag voor groen krijgt voordat de alternatieve signaalgroep groen geworden is. De fasecyclus toestand van de alternatieve signaalgroep wordt in dat geval “teruggezet” vanuit ROG() of RVG() naar WR().

De alternatieve realisatie van een signaalgroep wordt afgebroken indien dit nodig is om het startgroen moment van een verderop in de cyclus (fictief) conflicterende primaire signaalgroep te kunnen garanderen. De fasecyclus toestand van de alternatieve signaalgroep wordt in dat geval geforceerd naar GL().

4.3. Overslag primaire realisatie

Tijdens de alternatieve realisatie van een signaalgroep kan er tijdens de groenfase een moment ontstaan dat het voor een verderop in de cyclus (fictief) conflicterende primaire signaalgroep niet meer mogelijk is om groen te worden voor de start van zijn primaire gebied. Het primaire gebied wordt in dat geval versneld afgesloten.

5. Signaalgroep afwikkeling

5.1. Wachtstand groen

Voor iedere signaalgroep kan door middel van een schakelaar SCH(WGRn) gekozen worden tussen wachtstand rood en wachtstand groen.

Een wachtstand realisatie wordt toegekend indien geen enkele (fictief) conflicterende signaalgroep een aanvraag voor groen heeft en alle ontruimingstijden zijn verstreken. De signaalgroep maakt in dat geval de overgang van WR() naar ROG().
Aan signaalgroepen die op grond van een gedefinieerd uitstel moment (nog) niet groen mogen worden wordt nooit een wachtstand realisatie toegekend.

De fasecyclus toestand WG() wordt vervolgens aangehouden totdat een (fictief) conflicterende aanvraag ontstaat.
Een gedefinieerd afkap moment beëindigt eveneens de fasecyclus toestand WG().

5.2. Meeverleng groen

Voor iedere signaalgroep kan door middel van een schakelaar SCH(MVGn) gekozen worden tussen niet meeverlengen of meeverlengen totdat een (fictief) conflict gerealiseerd kan worden.
Een gedefinieerd afkap moment beëindigt eveneens de fasecyclus toestand MVG().

Signaalgroepen kunnen alleen meeverlengen indien voor tenminste één signaalgroep geldt:

R() && A() || G() && !WG() && (!MVG() || (H1e_loopt() || H2e_loopt())

5.3. Roodtijd bewaking

Voor het laten aanspreken van de fasebewaking kan gebruik worden gemaakt van de variabele KRA[]. In deze variabele wordt de actuele roodtijd na het ontstaan van een aanvraag voor groen opgeslagen.

Het ophogen van KRA[] wordt gehalteert tijdens fixatie en indien het stappenraster als gevolg synchronisatie wordt vertraagd. KRA[] wordt gereset bij de start van een stappenraster of indien voor een signaalgroep BL_OP() waar is.
KRA[] wordt ook gereset indien als gevolg van het synchronisatie mechanisme STAP onmiddellijk gelijk wordt gemaakt aan EPARM(SYNC_x).

5.4. Fixatie

Tijdens fixatie wordt voor iedere signaalgroep de fasecyclus overgang van MVG() naar GL() geblokkeerd.

5.5. Blokkeer optie

Door middel van BL_OP() kan de groensturing van een signaalgroep geblokkeerd worden. Indien BL_OP() waar is wordt de fasecyclus toestand “teruggezet” vanuit ROG() of RVG() naar WR() en wordt SCH(CPRIMn_x) altijd gelijk aan UIT verondersteld.

Zolang BL_OP() waar is zal de fasecyclus toestand WR() worden aangehouden en wordt een eventueel aanwezige A() door de CRSV module genegeerd.

5.6. Openhouden primair aanvraag gebied

Het primaire aanvraag gebied van een signaalgroep wordt automatisch afgesloten indien realisatie op het gedefinieerde startgroen moment niet meer mogelijk is.

Door het definiëren van een extra hulpfunctie PRM_APPL_SGn() kunnen aanvullende voorwaarden worden opgesteld waaraan moet worden voldaan voordat het primaire aanvraag gebied kan worden afgesloten.

Zolang de hulpfunctie PRM_APPL_SGn() waar is kan het primaire aanvraag gebied niet worden afgesloten. Zo nodig worden hiertoe alternatieve realisaties van (fictief) conflicterende richting geblokkeerd of afgebroken.

5.7. Blokkeer alternatieve realisatie

De CRSV module bepaalt automatisch voor iedere signaalgroep of er een alternatieve realisatie kan worden toegekend.

Door het definiëren van een extra hulpfunctie BAR_APPL_SGn() kunnen aanvullende voorwaarden worden opgesteld waaraan moet worden voldaan voordat een alternatieve realisatie kan worden toegekend.

Zolang de hulpfunctie BAR_APPL_SGn() waar is kan er geen alternatieve realisatie worden toegekend. Een actieve alternatieve realisatie wordt beëindigd.

5.8. Wachttijd voorspellers

Iedere signaalgroep kan door middel van één instructie in init_bij_inschakelen(void) gekoppeld worden aan een (31 led)wachttijdvoorspeller optioneel voorzien van een BUS sjabloon.

Signaalgroepen die gekoppeld zijn aan een wachttijdvoorspeller worden vanuit RVG() nooit teruggezet naar WR(). Voor de daadwerkelijke aansturing zijn de variabelen ALED[] en ABUS[] aanwezig.
De waarde van ALED[] geeft het aantal (resterende) leds die aangestuurd moeten worden. Het BUS sjabloon dient te worden aangestuurd indien ABUS[] is TRUE.

Definitie in de CRSV module:

De wachttijdvoorspellers worden als volgt gedefinieerd in init_bij_inschakelen(void):

```
#define WTV_BUS        /* Dit define opnemen indien de wachttijdvoorspellers voorzien zijn */
                       /* van een BUS sjabloon */

WTV_SG[n] = TRUE;      /* Door middel van dit statement wordt SGn gekoppeld aan een */
                       /* wachttijdvoorspeller */
```

6. Signaalgroep koppelingen

6.1. Voetgangersfiets koppeling

Een voetgangersfiets koppeling bestaat uit een voetgangersoversteek met één signaalgroep gekoppeld met maximaal twee parallelle fiets signaalgroepen. Voor het toepassen van deze koppeling gelden de volgende voorwaarden:

- De signaalgroepen hebben dezelfde (fictieve) conflicten.
- De uitstel momenten, EPARM(UITSTn_x), hebben dezelfde instelling.
- De startgroen momenten, EPARM(STARTn_x), hebben dezelfde instelling.
- Signaalgroepen mogen nooit onderdeel zijn van meerdere koppelingen.

De CRSV module handelt de voetgangersfiets koppeling als volgt af:

Algemeen:
De signaalgroepen maken, voor zover aangevraagd en niet reeds groen, gelijktijdig de overgang van RVG() naar G().

Indien voor één van de signaalgroepen geldt dat de fasecyclus toestand gelijk is aan ROG() of RVG() dan wordt voor de overige signaalgroepen de fasecyclus overgang van MVG() naar GL() geblokkeerd.

Meerealiseren en meeverlengen:
De fiets signaalgroepen realiseren- en verlengen mee met de voetganger signaalgroep.

Voor het mee realiseren van de fiets signaalgroepen onderling zijn twee schakelaars aanwezig, SCH(MREn1n2) en SCH(MREn2n1).

Definitie in de CRSV module:
De koppeling wordt als volgt gedefinieerd in init_bij_inschakelen(void):

```
proc_def_vtg_fts(SGvtg,SGfts1,SGfts2,MRfts1fts2,MRfts2fts1);

Met:    SGvtg          : Voetganger signaalgroep
        SGfts1         : 1e gekoppelde fiets signaalgroep
        SGfts2         : 2e gekoppelde fiets signaalgroep
        MRfts1fts2     : Meerealisatie van SGfts1 met SGfts2
        MRfts2fts1     : Meerealisatie van SGfts2 met SGfts1

Indien slechts één fietsrichting aanwezig dan als argument 3 t/m 5 “NIET” meegeven.
```

6.2. Progressieve voetgangersoversteek

Een progressieve voetgangersoversteek bestaat uit een voetgangersoversteek met twee voetganger signaalgroepen waarbij de “binnenste” lantaarns op de middenberm en de “buitenste” lantaarns op de beide zijbermen afzonderlijk geregeld worden. De oversteek kan gekoppeld worden met maximaal twee parallelle fiets signaalgroepen.

Voor het toepassen van deze koppeling gelden de volgende voorwaarden:

- De signaalgroepen hebben dezelfde (fictieve) conflicten.
- De uitstel momenten, EPARM(UITSTn_x), hebben dezelfde instelling.
- De startgroen momenten, EPARM(STARTn_x), hebben dezelfde instelling.
- Aan de voetganger signaalgroepen mag geen verlengfunctie worden toegekend.
- Signaalgroepen mogen nooit onderdeel zijn van meerdere koppelingen.

De CRSV module handelt de progressieve voetgangersoversteek als volgt af:

Algemeen:
De signaalgroepen maken, voor zover aangevraagd en niet reeds groen, gelijktijdig de overgang van RVG() naar G().

Indien voor één van de signaalgroepen geldt dat de fasecyclus toestand gelijk is aan ROG() of RVG() dan wordt voor de overige signaalgroepen de fasecyclus overgang van MVG() naar GL() geblokkeerd.

Meerealiseren en meeverlengen:
Een aanvraag van de “binnenste” voetganger signaalgroep wordt doorgeschreven naar de “buitenste” voetganger signaalgroep.

De fiets signaalgroepen realiseren- en verlengen mee met de voetganger signaalgroepen.

Voor het mee realiseren van de fiets signaalgroepen onderling zijn twee schakelaars aanwezig, SCH(MREn1n2) en SCH(MREn2n1).

Voetgangerskoppeling:
Koppeltijd TIJD(Tn1n2) (her)start tijdens het groen van de “binnenste” voetganger signaalgroep. Zolang TIJD(Tn1n2) waar is wordt voor de “buitenste” voetganger signaalgroep de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd. Indien de fasecyclus toestand van de “buitenste” voetganger signaalgroep gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().

Koppelgarantie:
Binnen de CRSV module zijn voor iedere progressieve voetgangersoversteek twee schakelaars aanwezig waarmee de koppelgarantie van deze voetgangerskoppeling kan worden bestuurd:

SCH(GKGAn1n2): Als deze schakelaar OP staat kunnen de signaalgroepen alternatief realiseren zonder de garantie dat de volledige voetgangerskoppeling kan worden gerealiseerd.

SCH(GKGPn1n2): Als deze schakelaar OP staat mogen (fictief) conflicterende prioriteitsingrepen de voetgangerskoppeling afbreken.

Definitie in de CRSV module:
De koppeling wordt als volgt gedefinieerd in init_bij_inschakelen(void):

```
proc_def_vtg_bb(SGvtg1,SGvtg2,Tvtg1vtg2,GKGAvtg1vtg2
               ,SGfts1,SGfts2,MRfts1fts2,MRfts2fts1,GKGPvtg1vtg2);

Met:    SGvtg1          : Voetganger signaalgroep binnenste lantaarns
        SGvtg2          : Voetganger signaalgroep buitenste lantaarns
        Tvtg1vtg2       : Koppeltijd van einde_G(SGvtg1) tot einde_G(SGvtg2)
        GKGAvtg1vtg2    : Geen koppelgarantie bij alternatieve realisatie
        SGfts1          : 1e gekoppelde fiets signaalgroep
        SGfts2          : 2e gekoppelde fiets signaalgroep
        MRfts1fts2      : Meerealisatie van SGfts1 met SGfts2
        MRfts2fts1      : Meerealisatie van SGfts2 met SGfts1
        GKGPvtg1vtg2    : Geen koppelgarantie bij conflicterende prioriteitsrealisatie

Indien geen fietsrichtingen aanwezig dan als argument 5 t/m 8 “NIET” meegeven
Indien slechts één fietsrichting aanwezig dan als argument 6 t/m 8 “NIET” meegeven.
```

6.2. Gescheiden voetgangersoversteek

Een gescheiden voetgangersoversteek bestaat uit een voetgangersoversteek met maximaal drie voetganger signaalgroepen die elk de oversteek voor één rijbaan verzorgen. De oversteek kan gekoppeld worden met maximaal twee parallelle fiets signaalgroepen, die ieder weer één volgrichting kunnen hebben. De oversteek bestaat dus maximaal uit zeven signaalgroepen.

De conflicten van de voetganger signaalgroepen worden onderling fictief doorgeschreven. Als er voor een fiets signaalgroep een volgrichting is gedefinieerd dan worden ook hier de conflicten onderling fictief doorgeschreven zodat de voedende- en de volgrichting dezelfde (fictieve) conflicten hebben.

De fictieve conflicten worden door de CRSV module bepaald en dienen dus niet in craptab.c gedefinieerd te worden.

Voor het toepassen van deze koppeling gelden de volgende voorwaarden:

- De signaalgroepen hebben dezelfde (fictieve) conflicten.
- De uitstel momenten, EPARM(UITSTn_x), hebben dezelfde instelling.
- De startgroen momenten, EPARM(STARTn_x), hebben dezelfde instelling.
- Aan de voetganger signaalgroepen mag geen verlengfunctie worden toegekend.
- Signaalgroepen mogen nooit onderdeel zijn van meerdere koppelingen.

De CRSV module handelt de gescheiden voetgangersoversteek als volgt af:

Algemeen:
De signaalgroepen maken, voor zover aangevraagd en niet reeds groen, gelijktijdig de overgang van RVG() naar G().
Indien er geen parallelle fietsrichtingen zijn gedefinieerd en er is slechts aan één buitenzijde aangevraagd dan mag de voedende voetganger al groen worden tijdens het aflopen van de ontruimingstijden naar zijn volgrichting.

Indien voor één van de signaalgroepen geldt dat de fasecyclus toestand gelijk is aan ROG() of RVG() dan wordt voor de overige signaalgroepen de fasecyclus overgang van MVG() naar GL() geblokkeerd.

Meerealiseren en meeverlengen:
Een aanvraag aan de buitenzijde wordt doorgeschreven naar de volgrichting.

De voedende fiets signaalgroepen realiseren- en verlengen mee met de voetganger signaalgroepen.

Voor het mee realiseren van de voedende fiets signaalgroepen onderling zijn twee schakelaars aanwezig, SCH(MREn1n2) en SCH(MREn2n1).

Voor het mee realiseren van de voedende fiets signaalgroep met zijn eigen volgrichting is één schakelaar aanwezig, SCH(MREn1n2).

Voetgangerskoppeling:
Koppeltijd TIJD(Tn1n2) (her)start op begin_G(n1) indien signaalgroep SG(n1) aan de buitenzijde is aangevraagd. Zolang TIJD(Tn1n2) waar is wordt voor de volgrichting

de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd. Indien de fasecyclus toestand van de volgrichting gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().

Indien drie voetganger signaalgroepen zijn gedefinieerd (her)starten er twee koppeltijden op begin_G(n1) of begin_G(n2) indien aan de buitenkant gedrukt is.

Indien in geval van drie voetganger signaalgroepen de “middelste” signaalgroep wordt aangevraagd (her)start op begin_G(n3) een koppeltijd naar één of twee van de overige signaalgroepen. (dit is dan afhankelijk van waar gedrukt is)

Fietskoppeling:
Koppeltijd TIJD(TBn1n2) (her)start op begin_G(n1). Koppeltijd TIJD(TEn1n2) (her)start tijdens G(n1) onder de voorwaarde dat HF(HNLn1n2) waar is. Zolang TIJD(TBn1n2) of TIJD(TEn1n2) waar is wordt voor de volgrichting de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd. Indien de fasecyclus toestand van de volgrichting gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().
Door het gebruik van hulpfunctie HF(HNLn1n2) zijn er eenvoudig aanvullende voorwaarden te programmeren voor het (her)starten van koppeltijd TIJD(TEn1n2).

Koppelgarantie:
Binnen de CRSV module zijn voor iedere gescheiden voetgangersoversteek twee schakelaars aanwezig waarmee de koppelgarantie van deze voetgangerskoppeling kan worden bestuurd:

SCH(GKGAn1n2): Als deze schakelaar OP staat kunnen de signaalgroepen alternatief realiseren zonder de garantie dat de volledige voetgangerskoppeling kan worden gerealiseerd.

SCH(GKGPn1n2): Als deze schakelaar OP staat mogen (fictief) conflicterende prioriteitsingrepen de voetgangerskoppeling afbreken.

Fietskoppelingen worden altijd gegarandeerd.

Definitie in de CRSV module:

De koppeling wordt als volgt gedefinieerd in init_bij_inschakelen(void):

```
proc_def_vtg_go(SGvtg1,SGvtg2,SGvtg1_b,SGvtg2_b,Tvtg1vtg2,Tvtg2vtg1,GKGAvtg1vtg2
               ,SGfts1,SGfts1v,MRfts1fts1v,TBfts1fts1v,TEfts1fts1v,HNLfts1fts1v
               ,SGfts2,SGfts2v,MRfts2fts2v,TBfts2fts2v,TEfts2fts2v,HNLfts2fts2v
               ,MRfts1fts2,MRfts2fts1,GKGPvtg1vtg2,SGvtg3,SGvtg3_31,SGvtg3_32,Tvtg1vtg3
               ,Tvtg2vtg3,Tvtg3vtg1,Tvtg3vtg2);

Met:    SGvtg1          : 1e Voetganger signaalgroep
        SGvtg2          : 2e Voetganger signaalgroep
        SGvtg1_b        : Buitenste drukknop SGvtg1
        SGvtg2_b        : Buitenste drukknop SGvtg2
        Tvtg1vtg2       : Koppeltijd van begin_G(SGvtg1) tot einde_G(SGvtg2)
        Tvtg2vtg1       : Koppeltijd van begin_G(SGvtg2) tot einde_G(SGvtg1)
        GKGAvtg1vtg2    : Geen koppelgarantie bij alternatieve realisatie
        SGfts1          : 1e gekoppelde fiets signaalgroep
        SGfts1v         : Volgrichting SGfts1
        MRfts1fts1v     : Meerealisatie van SGfts1 met SGfts1v
        TBfts1fts1v     : Koppeltijd van begin_G(SGfts1) tot einde_G(SGfts1v)
        TEfts1fts1v     : Koppeltijd van einde_G(SGfts1) tot einde_G(SGfts1v)
        HNLfts1fts1v    : Hulpfunctie (her)start koppeltijd TEfts1fts1v
        SGfts2          : 2e gekoppelde fiets signaalgroep
        SGfts2v         : Volgrichting SGfts2
        MRfts2fts2v     : Meerealisatie van SGfts2 met SGfts2v
        TBfts2fts2v     : Koppeltijd van begin_G(SGfts2) tot einde_G(SGfts2v)
        TEfts2fts2v     : Koppeltijd van einde_G(SGfts2) tot einde_G(SGfts2v)
        HNLfts2fts2v    : Hulpfunctie (her)start koppeltijd TEfts2fts2v
        MRfts1fts2      : Meerealisatie van SGfts1 met SGfts2
        MRfts2fts1      : Meerealisatie van SGfts2 met SGfts1
        GKGPvtg1vtg2    : Geen koppelgarantie bij conflicterende prioriteitsrealisatie
        SGvtg3          : 3e voetganger fasecyclus - dit is de "middelste" fasecyclus
        SGvtg3_31       : Drukknop FCvtg3 -> FCvtg1
        SGvtg3_32       : Drukknop FCvtg3 -> FCvtg2
        Tvtg1vtg3       : Koppeltijd van begin_G(SGvtg1) tot einde_G(SGvtg3)
        Tvtg2vtg3       : Koppeltijd van begin_G(SGvtg2) tot einde_G(SGvtg3)
        Tvtg3vtg1       : Koppeltijd van begin_G(SGvtg3) tot einde_G(SGvtg1)
        Tvtg3vtg2       : Koppeltijd van begin_G(SGvtg3) tot einde_G(SGvtg2)

Indien geen fietsrichtingen aanwezig dan als argument 8 t/m 21 "NIET" meegeven
Indien volgrichting SGfts1 niet aanwezig dan als argument  9 t/m 13 "NIET" meegeven.
Indien volgrichting SGfts2 niet aanwezig dan als argument 15 t/m 19 "NIET" meegeven.
Indien slecht één fietsrichting aanwezig dan als argument 14 t/m 21 "NIET" meegeven.

Indien geen 3e voetganger aanwezig dan als argument 23 t/m 29 "NIET" meegeven.
```

6.4. Seriële koppeling

Een seriële koppeling bestaat uit maximaal drie signaalgroepen waarbij de groenfase van een voedende richting altijd binnen een instelbare tijd gevolgd wordt door een groenfase van zijn volgrichting(en).

Voor het toepassen van deze koppeling gelden de volgende voorwaarden:

- Het einde primair moment, EPARM(EINDEn_x), van een voedende richting moet voor het einde primair moment van zijn volgrichting(en) liggen.
- Signaalgroepen mogen onderdeel zijn van meerdere seriële koppelingen, tenzij hierdoor een koppeling ontstaat van vier of meerdere signaalgroepen.

De wijze van afhandeling binnen de CRSV module wordt hieronder beschreven voor de koppeling van SG(n1) -> SG(n2). Bij drie signaalgroepen gelden dezelfde voorwaarden voor de koppeling van SG(n2) -> SG(n3).

Algemeen:
Indien een voedende richting de overgang maakt van RVG() naar G() dan zal zijn directe volgrichting na maximaal TIJD(TVn1n2) seconden groen zijn.
Voor de primaire realisatie is deze garantie op groen alleen aanwezig indien het start primair moment van de directe volgrichting maximaal TIJD(TVn1n2) seconden na het start primair moment van de voedende richting is gedefinieerd.

Indien voor een voedende richting geldt dat de fasecyclus toestand gelijk is aan ROG() of RVG() dan wordt voor zijn directe volgrichting de overgang van MVG() naar GL() geblokkeerd.

Meerealiseren:
Een aanvraag van een voedende richting wordt op begin_G() doorgeschreven naar zijn directe volgrichting.

Koppeling:
Koppeltijd TIJD(TBn1n2) (her)start op begin_G(n1).

Koppeltijd TIJD(TEn1n2) (her)start tijdens G(n1) onder de voorwaarde dat één van de koplussen van de voedende richting bezet is.
Indien er geen koplussen gedefinieerd zijn (her)start koppeltijd TIJD(TEn1n2) altijd tijdens G(n1).

Koppeltijd TIJD(TMn1n2) (her)start zolang TIJD(TBn1n2) of TIJD(TEn1n2) waar is. TIJD(TMn1n2) wordt afgebroken indien H2e_TIJD(n2) niet meer waar is.
TIJD(TMn1n2) is bedoeld om “langzame” voertuigen alsnog over de stopstreep van de directe volgrichting te krijgen.

Zolang TIJD(TBn1n2), TIJD(TEn1n2) of TIJD(TMn1n2) waar is wordt voor de directe volgrichting de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd. Indien de fasecyclus toestand van de directe volgrichting gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().

De koppeling is schakelbaar door middel van HF(HKI_n1n2). Het daadwerkelijk inschakelen van een koppeling wordt daarbij uitgesteld totdat alle voedende richtingen geen groen meer tonen.
(wachtstand- en meeverlenggroen voorwaarden worden hiervoor zonodig tijdelijk uitgeschakeld)

Het uitschakelen van een koppeling wordt uitgesteld totdat alle nalooptijden zijn verstreken.

Definitie in de CRSV module:

De koppeling wordt als volgt gedefinieerd in init_bij_inschakelen(void):

```
set_HF(HKIn1n2);       /* Indien koppeling aktief bij inschakelen regelprogramma */
reset_HF(HKIn1n2);     /* Indien koppeling NIET aktief bij inschakelen regelprogramma */

proc_def_hki(SGn1,SGn2,TVn1n2,TBn1n2,TEn1n2,TMn1n2,SGn1_k1,SGn1_k2,SGn1_k3,
                  SGn3,TVn2n3,TBn2n3,TEn2n3,TMn2n3,SGn2_k1,SGn2_k2,SGn2_k3,HKI_n1n2);

Met:    SGn1            : Voedende signaalgroep
        SGn2            : 1e volg signaalgroep
        TVn1n2          : Maximale voorstart signaalgroep SGn1 t.o.v. signaalgroep SGn2
        TBn1n2          : Koppeltijd van begin_G(SGn1) tot einde_G(SGn2)
        TEn1n2          : Koppeltijd van einde_G(SGn1) tot einde_G(SGn2)
        TMn1n2          : Verlengtijd SGn2 na afloop koppeltijden TBn1n2 en TEn1n2
        SGn1_k1         : Definitie 1e koplus SGn1
        SGn1_k2         : Definitie 2e koplus SGn1
        SGn1_k3         : Definitie 3e koplus SGn1
        SGn3            : 2e volg signaalgroep
        TVn2n3          : Maximale voorstart signaalgroep SGn2 t.o.v. signaalgroep SGn3
        TBn2n3          : Koppeltijd van begin_G(SGn2) tot einde_G(SGn3)
        TEn2n3          : Koppeltijd van einde_G(SGn2) tot einde_G(SGn3)
        TMn2n3          : Verlengtijd SGn3 na afloop koppeltijden TBn2n3 en TEn2n3
        SGn2_k1         : Definitie 1e koplus SGn2
        SGn2_k2         : Definitie 2e koplus SGn2
        SGn2_k3         : Definitie 3e koplus SGn2
        HKI_n1n2        : Hulpfunctie koppeling aktief

Indien koplus(sen) niet aanwezig dan als argument “NIET” meegeven.
Indien de 2e volgrichting niet aanwezig is dan als argument 10 t/m 17 “NIET” meegeven.
```

7. Beïnvloeding door openbaar vervoer

7.1. Randvoorwaarden en uitgangspunten

Binnen de CRSV module is een beperkte beïnvloeding door openbaar vervoer mogelijk. De vaste (netwerk) cyclustijd waarbinnen alle signaalgroepen, voor zover aangevraagd, gerealiseerd moeten worden is hierbij de belangrijkste beperkende factor.

Voor de prioriteit van het openbaar vervoer gelden de volgende randvoorwaarden:

- Primaire realisaties van (fictief) conflicterende signaalgroepen mogen niet worden overgeslagen, wel mogen deze realisaties worden ingekort. Dit inkorten is mogelijk door:
    - De start van het groen uit te stellen.
    - De groenfase af te breken.
    - De groenfase te doorsnijden.
- Seriële koppelingen blijven gegarandeerd.
- Parallelle koppelingen voor langzaam verkeer blijven gegarandeerd.
  Voor voetgangerskoppelingen is deze garantie schakelbaar door middel van SCH(GKGPn1n2).

Voor iedere signaalgroep is door middel van parameter EPARM(ABMOVn_x) per stappenraster instelbaar hoe de groenfase mag worden ingekort door een (fictief) conflicterende prioriteitsingreep.
EPARM(ABMOVn_x) is een optelsom van de toegestane inkort methoden. 1.0 = uitstellen toegestaan; 2.0 = afbreken toegestaan; 4.0 = doorsnijden toegestaan. EPARM(ABMOVn_x) = 7.0 betekent dus dat alle inkort methoden zijn toegestaan.

Voor iedere signaalgroep is door middel van parameter EPARM(KAPOVn) het ondermaximum voor inkorten instelbaar als percentage van de primaire groenduur.
De primaire groenduur wordt bepaald door het verschil tussen EPARM(STARTn_x) en EPARM(EINDEn_x).

Voor iedere signaalgroep is door middel van parameter EPARM(TPMOVn_x) per stappenraster instelbaar of de prioriteit mag starten voor het moment einde uitstel of langer mag duren dan het afkap moment.
EPARM(TPMOVn_x) is een opstelsom van de toegestane prioriteit methoden. 1.0 = starten voor einde uitstel toegestaan; 2.0 = aanhouden groen na afkap moment toegestaan. EPARM(TPMOVn_x) = 3.0 betekent dus dat prioriteit altijd is toegestaan.

Indien prioriteit voor het moment einde uitstel is toegestaan dan is de maximale tijd voor het moment van einde uitstel apart instelbaar door middel van parameter EPARM(PVEUn).

7.2. Toekennen prioriteitsrealisaties

Aan ieder voertuig kan als volgt geconditioneerde prioriteit worden toegekend:

- Niveau 0: Geen prioriteit.
- Niveau 1: Alleen aanhouden eigen groenfase toegestaan.
- Niveau 2: Prioriteitsrealisatie (en aanhouden eigen groenfase) toegestaan.

Voor de afwikkeling van de prioriteit zijn daarnaast per voertuig de volgende gegevens relevant:

- Vrije afstand rijtijd tot de stopstreep.
  Dit is de verwachte rijtijd vanaf de inmelding tot de stopstreep er vanuit gaande dat het voertuig vrij kan afrijden.
- Minimum groenduur voor stopstreep passage.
  Dit is de minimale groenduur die verstreken moet zijn voordat het voertuig de stopstreep passeert. Deze tijd is bedoeld om het verkeer in de wachtrij voor het ingemelde voertuig te laten afrijden.

De CRSV module rekent aan de hand van deze gegevens uit of het een “gat” in het stappenraster kan creëren dat groot genoeg is om het ingemelde voertuig te laten passeren. Dit kan door het afbreken, uitstellen en/of doorsnijden van (fictief) conflicterende signaalgroepen eventueel in combinatie met het ontbreken van (fictief) conflicterende primaire aanvragen.

Als dit het geval is en aan het voertuig is prioriteitsniveau 2 toegekend (of prioriteitsniveau 1 indien de betreffende signaalgroep groen is) dan zal de CRSV module prioriteit toekennen aan het betreffende voertuig.

Voor een signaalgroep waaraan een prioriteitsrealisatie is toegekend wordt uitgerekend wat het gewenste startgroen moment is. Afhankelijk van dit moment worden (fictief) conflicterende signaalgroepen zo nodig vanuit ROG() of RVG() “teruggezet” naar WR() of wordt de fasecyclus overgang naar GL() geforceerd indien de (fictief) conflicterende signaalgroep groen is.

De signaalgroep waaraan de prioriteitsrealisatie is toegekend maakt de overgang van WR() naar ROG(). Zolang het betreffende voertuig aanwezig is wordt de fasecyclus overgang van VAG2e() naar MVG() geblokkeerd. Indien de fasecyclus toestand gelijk is aan MVG() wordt deze “teruggezet” naar VAG2e().

Tijdens de prioriteitsrealisatie van een signaalgroep kan er tijdens de groenfase een moment ontstaan dat het voor een verderop in de cyclus (fictief) conflicterende primaire signaalgroep niet meer mogelijk is om groen te worden. Het primaire gebied wordt in dat geval versneld afgesloten.
Niet meer mogelijk om groen te worden betekent hier dat er onvoldoende ruimte is voor het ingestelde ondermaximum.

Voor iedere in- en uitmelding kunnen in crapcod.c per voertuig de volgende gegevens worden meegegeven aan de CRSV module:

- Prioriteitsniveau.
- Vrije afstand rijtijd tot de stopstreep in seconden.
- Minimum duur groenfase voor (vrije) stopstreep passage in seconden.
- Voertuignummer.

Bij het ontbreken van deze gegevens worden door de CRSV module de volgende default instellingen gebruikt:

- EPARM(DTTSn) default vrije afstand rijtijd tot de stopstreep in seconden.
- EPARM(DVRGn) default minimum duur groenfase voor (vrije) stopstreep passage in seconden.

De default instelling voor het prioriteitsniveau is altijd NUL. (geen prioriteit)
De default instelling voor het voertuignummer is altijd NUL. (voertuignummer is onbekend)

Als bij een inmelding blijkt dat het betreffende voertuig reeds was ingemeld dan worden de gegevens van het betreffende voertuig ge-update. Op deze wijze kan bijvoorbeeld de verwachte tijd tot stopstreeppassage of het prioriteitsniveau worden bijgesteld.

7.3. Uitmeld bewaking

Een voertuig dat zich niet uitmeldt zal de groenfase verder verlengen analoog aan reguliere voertuigafhankelijke regelingen. De uitmeld bewaking spreekt aan indien:

Het voertuig langer aanwezig is dan de ingestelde duur uitmeld bewaking, EPARM(UITBn).

EN

Het voertuig langer aanwezig is dan de vrije afstand rijtijd tot de stopstreep. **EN**

- De betreffende signaalgroep langer groen is dan de ingestelde duur uitmeld bewaking, EPARM(UITBn) **OF**
- De fasecyclus toestand van de betreffende signaalgroep is VG[ ] en H2e_loopt() is niet meer waar. (er zit geen verkeer meer voor het voertuig)

Een uitgestelde primaire (fictieve) conflictrichting mag hierdoor niet worden “weggedrukt”. Dit stelt dus een maximum aan de duur van de groenverlenging als gevolg van het ontbreken van een uitmelding.

Voertuigen waaraan prioriteit is toegekend en waarvan de uitmelding niet binnenkomt worden altijd (automatisch) uitgemeld (aanspreken uitmeld bewaking) op begin_R().

7.4. Definitie in de CRSV module

De OV beïnvloeding wordt als volgt gedefinieerd in init_bij_inschakelen(void):

```
proc_def_ov(SGov,DTTSov,DVGRov,UITBov,PVEUov);

Met:    SGov           : Signaalgroep met OV beïnvloeding
        DTTSov         : Default vrije afstand rijtijd tot de stopstreep
        DVGRov         : Default minimum duur groenfase voor (vrije) stopstreeppassage
        UITBov         : Uitmeldbewaking
        PVEUov         : Tijd voor einde uitstel waarbij prioriteit toegestaan

DTTSov, DVGRov, UITBov en PVEUov zijn gedefinieerd als EPARM().
Indien uitstelmoment niet aanwezig dan voor PVEUov als argument “NIET” meegeven.
```

8. Programma technische beschrijving

8.1. Koppeling CRSV module met de C-regelaar

Vanaf versie 5.0 is het mogelijk om de CRSV module te combineren met een RWS-C "blokken" regeling. Als van deze combinatie gebruik wordt gemaakt dient het volgende include statement te worden opgenomen in craptab.c:
#include "CRSV_CRAPTAB.H"

Indien gebruik wordt gemaakt van de oude niet geintergreerde versie van RWS-C dient het volgende define statement te worden opgenomen in crapcod.c:
#define RWSC_old
LET OP: Dit define moet geplaatst worden voor de include van crsv.c.

De CRSV module maakt verder gebruik van een aantal signaalgroep opties. Indien gebruik wordt gemaakt van een combinatie met een RWS-C "blokken" regeling moet de aansturing vanuit de CRSV module als volgt worden opgenomen:

MA-optie:

```
/*     SG gebonden optie:       MA_OP(SGn)                          */
Bool   MA_funct_02(void)        { return CRSV_TO_CR[SG02][MA_CRSV]; }
Bool   MA_funct_05(void)        { return CRSV_TO_CR[SG05][MA_CRSV]; }
Bool   MA_funct_08(void)        { return CRSV_TO_CR[SG08][MA_CRSV]; }

struct SG_OPTIE                 MA_OP_tabel[] = {
/*     optie voor sg:           inhoud van de optie:                */
/*     SGn,                     functie,                            */
       SG02,                    MA_funct_02,
       SG05,                    MA_funct_05,
       SG08,                    MA_funct_08,
       STOP};
```

PPS-optie:

```
/*     SG gebonden optie:       PPS_OP(SGn)                         */
Bool   PPS_funct_02(void)       { return CRSV_TO_CR[SG02][PPS_CRSV]; }
Bool   PPS_funct_05(void)       { return CRSV_TO_CR[SG05][PPS_CRSV]; }
Bool   PPS_funct_08(void)       { return CRSV_TO_CR[SG08][PPS_CRSV]; }

struct SG_OPTIE                 PPS_OP_tabel[] = {
/*     optie voor sg:           inhoud van de optie:                */
/*     SGn,                     functie,                            */
       SG02,                    PPS_funct_02,
       SG05,                    PPS_funct_05,
       SG08,                    PPS_funct_08,
       STOP};
```

RWR-optie:

```
/*     SG gebonden optie:       RWR_OP(SGn)                         */
Bool   RWR_funct_02(void)       { return CRSV_TO_CR[SG02][RWR_CRSV]; }
Bool   RWR_funct_05(void)       { return CRSV_TO_CR[SG05][RWR_CRSV]; }
Bool   RWR_funct_08(void)       { return CRSV_TO_CR[SG08][RWR_CRSV]; }

struct SG_OPTIE                 RWR_OP_tabel[] = {
/*     optie voor sg:           inhoud van de optie:                */
/*     SGn,                     functie,                            */
       SG02,                    RWR_funct_02,
       SG05,                    RWR_funct_05,
       SG08,                    RWR_funct_08,
       STOP};
```

VRVG-optie:

```
/*      SG gebonden optie:        VRVG_OP(SGn)                          */
Bool    VRVG_funct_02(void)       { return CRSV_TO_CR[SG02][VRVG_CRSV]; }
Bool    VRVG_funct_05(void)       { return CRSV_TO_CR[SG05][VRVG_CRSV]; }
Bool    VRVG_funct_08(void)       { return CRSV_TO_CR[SG08][VRVG_CRSV]; }

struct SG_OPTIE                   VRVG_OP_tabel[] = {
/*      optie voor sg:            inhoud van de optie:                  */
/*      SGn,                      functie,                              */
        SG02,                     VRVG_funct_02,
        SG05,                     VRVG_funct_05,
        SG08,                     VRVG_funct_08,
        STOP};
```

VVAG2e-optie:

```
/*      SG gebonden optie:        VVAG2e_OP(SGn)                        */
Bool    VVAG2e_funct_02(void)     { return CRSV_TO_CR[SG02][VVAG2e_CRSV]; }
Bool    VVAG2e_funct_05(void)     { return CRSV_TO_CR[SG05][VVAG2e_CRSV]; }
Bool    VVAG2e_funct_08(void)     { return CRSV_TO_CR[SG08][VVAG2e_CRSV]; }

struct SG_OPTIE                   VVAG2e_OP_tabel[] = {
/*      optie voor sg:            inhoud van de optie:                  */
/*      SGn,                      functie,                              */
        SG02,                     VVAG2e_funct_02,
        SG05,                     VVAG2e_funct_05,
        SG08,                     VVAG2e_funct_08,
        STOP};
```

RVAG2e-optie:

```
/*      SG gebonden optie:        RVAG2e_OP(SGn)                        */
Bool    RVAG2e_funct_02(void)     { return CRSV_TO_CR[SG02][RVAG2e_CRSV]; }
Bool    RVAG2e_funct_05(void)     { return CRSV_TO_CR[SG05][RVAG2e_CRSV]; }
Bool    RVAG2e_funct_08(void)     { return CRSV_TO_CR[SG08][RVAG2e_CRSV]; }

struct SG_OPTIE                   RVAG2e_OP_tabel[] = {
/*      optie voor sg:            inhoud van de optie:                  */
/*      SGn,                      functie,                              */
        SG02,                     RVAG2e_funct_02,
        SG05,                     RVAG2e_funct_05,
        SG08,                     RVAG2e_funct_08,
        STOP};
```

VMVG-optie:

```
/*      SG gebonden optie:        VMVG_OP(SGn)                          */
Bool    VMVG_funct_02(void)       { return CRSV_TO_CR[SG02][VMVG_CRSV]; }
Bool    VMVG_funct_05(void)       { return CRSV_TO_CR[SG05][VMVG_CRSV]; }
Bool    VMVG_funct_08(void)       { return CRSV_TO_CR[SG08][VMVG_CRSV]; }

struct SG_OPTIE                   VMVG_OP_tabel[] = {
/*      optie voor sg:            inhoud van de optie:                  */
/*      SGn,                      functie,                              */
        SG02,                     VMVG_funct_02,
        SG05,                     VMVG_funct_05,
        SG08,                     VMVG_funct_08,
        STOP};
```

AFK-optie:

```
/*      SG gebonden optie:        AFK_OP(SGn)                           */
Bool    AFK_funct_02(void)        { return CRSV_TO_CR[SG02][AFK_CRSV]; }
Bool    AFK_funct_05(void)        { return CRSV_TO_CR[SG05][AFK_CRSV]; }
Bool    AFK_funct_08(void)        { return CRSV_TO_CR[SG08][AFK_CRSV]; }

struct SG_OPTIE                   AFK_OP_tabel[] = {
/*      optie voor sg:            inhoud van de optie:                  */
/*      SGn,                      functie,                              */
        SG02,                     AFK_funct_02,
        SG05,                     AFK_funct_05,
        SG08,                     AFK_funct_08,
        STOP};
```

8.2. Parameters craptab.c

De CRSV module maakt gebruik van de volgende regelparameters welke in craptab.c gedefinieerd worden.

Schakelaars:
De volgende schakelaars zijn aanwezig in de CRSV module. De declaratie is echter niet verplicht. Bij de functie aanroepen kan gebruik worden gemaakt van de schakelaars SCH(NOOIT) en SCH(ALTIJD).

```
/*     SCHAKELAARtabel                                          */
struct SCHAKELAAR SCHAKELAAR_tabel[] = {
/*     SCH():                  instelling:                      */
/*     NAAM,                   AAN (of UIT),                    */

/*     Cyclische (primaire) realisatie                          */
/*     -------------------------------                          */
       CPRIM02,                UIT,            /* signaalgroep 02 */
       CPRIM05,                UIT,            /* signaalgroep 05 */
       CPRIM08,                UIT,            /* signaalgroep 08 */
       CPRIM11,                UIT,            /* signaalgroep 11 */

/*     Wachtstand groen                                         */
/*     ----------------                                         */
       WGR02,                  UIT,            /* signaalgroep 02 */
       WGR05,                  UIT,            /* signaalgroep 05 */
       WGR08,                  UIT,            /* signaalgroep 08 */
       WGR11,                  UIT,            /* signaalgroep 11 */

/*     MEEverlengen tot conflict gerealiseerd kan worden        */
/*     -------------------------------------------------        */
       MVG02,                  AAN,            /* signaalgroep 02 */
       MVG05,                  AAN,            /* signaalgroep 05 */
       MVG08,                  AAN,            /* signaalgroep 08 */
       MVG11,                  AAN,            /* signaalgroep 11 */

/*     Geen koppelgarantie bij alternatieve realisatie          */
/*     -----------------------------------------------          */
       GKGA3132,               UIT,       /* signaalgroep 31 en 32 */
       GKGA3334,               UIT,       /* signaalgroep 33 en 34 */

/*     Geen koppelgarantie bij conflicterende prioriteitsrealisatie */
/*     ------------------------------------------------------------ */
       GKGP3132,               UIT,       /* signaalgroep 31 en 32 */
       GKGP3334,               UIT,       /* signaalgroep 33 en 34 */

/*     MEErealisaties                                           */
/*     --------------                                           */
       MR2122,                 AAN,      /* signaalgroep 21 met 22 */
       MR2221,                 AAN,      /* signaalgroep 22 met 21 */

/*     Overige schakelaars                                      */
/*     -------------------                                      */
       NOOIT,                  UIT, /* Hulpschalelaar altijd "UIT" */
       ALTIJD,                 AAN, /* Hulpschakelaar altijd "AAN" */

       STOP};
```

Tijdelementen:
De volgende tijdelementen zijn aanwezig in de CRSV module:

```
/*     TIJDEN tabel                                                    */
struct TIJDEN           TIJDEN_tabel[] = {
/*     TIJD():          instelling:                                    */
/*     SGn_i,            x.y,                                          */

/*     Koppeltijden progressieve voetgangersoversteek                  */
/*     ----------------------------------------------                  */
       T3292,            6.0, /* einde_G(SG32) -> einde_G(SG92)       */

/*     Koppeltijden gescheiden voetgangersoversteek                    */
/*     --------------------------------------------                    */
       T3334,           15.0, /* begin_G(SG33) -> einde_G(SG34)       */
       T3433,           15.0, /* begin_G(SG34) -> einde_G(SG33)       */

/*     Koppeltijden fietskoppeling                                     */
/*     ---------------------------                                     */
       TB2383,           5.0, /* begin_G(SG23) -> einde_G(SG83)       */
       TE2383,           3.0, /* einde_G(SG23) -> einde_G(SG83)       */

       TB2484,           5.0, /* begin_G(SG24) -> einde_G(SG84)       */
       TE2484,           3.0, /* einde_G(SG24) -> einde_G(SG84)       */

/*     Koppeltijden seriele koppeling                                  */
/*     ------------------------------                                  */
       TV0262,          10.0, /* max.voorstart SG02 tov begin_G(SG62) */
       TB0262,          15.0, /* begin_G(SG02) -> einde_G(SG62)       */
       TE0262,          10.0, /* einde_G(SG02) -> einde_G(SG62)       */
       TM0262,           5.0, /* max.VA groen SG62 na afloop koppeltijden */

       TV6272,          10.0, /* max.voorstart SG62 tov begin_G(SG72) */
       TB6272,          15.0, /* begin_G(SG62) -> einde_G(SG72)       */
       TE6272,          10.0, /* einde_G(SG62) -> einde_G(SG72)       */
       TM6272,           5.0, /* max.VA groen SG72 na afloop koppeltijden */

       TV0868,          10.0, /* max.voorstart SG08 tov begin_G(SG68) */
       TB0868,          15.0, /* begin_G(SG08) -> einde_G(SG68)       */
       TE0868,          10.0, /* einde_G(SG08) -> einde_G(SG68)       */
       TM0868,           5.0, /* max.VA groen SG68 na afloop koppeltijden */

/*     Overige tijdelementen                                           */
/*     ---------------------                                           */
       T_OMSC,           1.0,    /* tijdvertraging omschakelen        */
       T_SYNC,           2.0,    /* duur synchronisatie signaal       */
                                 /* ... alleen in complexcoördinator  */
       STOP};
```

Extra Geheel Getal Parameters:
De volgende extra geheel getal parameters zijn aanwezig in de CRSV module:

```
/*     extra PARMS tabel                                               */
struct EGGPARMS         EGGPARMS_tabel[] = {
/*     EGGPARM(),       instelling                                     */

/*     CRSV versie                                                     */
/*     -----------                                                     */
       EG_CRSV_VERSIE,  60,      /* CRSV versie                       */

       STOP};
```

Extra parameters:
De volgende extra parameters zijn aanwezig in de CRSV module:

```
/*     extra PARMS tabel                                                */
struct EPARMS       EPARMS_tabel[] = {
/*     EPARM(),     instelling                                          */

/*     Bijzondere programma keuze                                       */
/*     -------------------------                                       */
/*     1.0 : LOKAAL bedrijf                                            */
/*     1.1 : LOKAAL bedrijf met versneld doorstappen                   */
/*     2.0 : CRSV =(default)                                           */
/*     3.0 : STAR   bedrijf                                            */
       PROGKEUZE,            2.0,         /* bijz. programma keuze     */

/*     Default stappenraster                                            */
/*     ---------------------                                           */
       DEFPRG,               1.0,         /* default stappenraster     */

/*     Stappenraster 1                                                  */
/*     ---------------                                                 */
       CTYD_1,             103.0,         /* cyclustijd                */
       INSC_1,               1.0,         /* inschakelpunt             */
       OMSC_1,               0.0,         /* omschakelpunt             */
       SYNC_1,               3.0,         /* synchronisatiepunt        */

/*     Stappenraster 1 -> signaalgroep 02                               */
/*     ----------------------------------                              */
       UITST02_1,            0.0,         /* einde uitstel             */
       START02_1,           56.0,         /* start primair             */
       WACHT02_1,            0.0,         /* start verlengen           */
       EINDE02_1,           85.0,         /* einde primair             */
       AFKAP02_1,            0.0,         /* einde groen               */

/*     Stappenraster 1 -> signaalgroep 05                               */
/*     ----------------------------------                              */
       START05_1,           19.0,         /* start primair             */
       EINDE05_1,           33.0,         /* einde primair             */

/*     Stappenraster 2                                                  */
/*     ---------------                                                 */
       CTYD_2,             103.0,         /* cyclustijd                */
       INSC_2,               1.0,         /* inschakelpunt             */
       OMSC_2,               0.0,         /* omschakelpunt             */
       SYNC_2,               3.0,         /* synchronisatiepunt        */

/*     Stappenraster 2 -> signaalgroep 02                               */
/*     ----------------------------------                              */
       UITST02_2,            0.0,         /* einde uitstel             */
       START02_2,           56.0,         /* start primair             */
       WACHT02_2,            0.0,         /* start verlengen           */
       EINDE02_2,           85.0,         /* einde primair             */
       AFKAP02_2,            0.0,         /* einde groen               */

/*     Stappenraster 2 -> signaalgroep 05                               */
/*     ----------------------------------                              */
       START05_2,           19.0,         /* start primair             */
       EINDE05_2,           33.0,         /* einde primair             */
```

De declaratie van UITSTn_x, WACHTn_x en AFKAPn_x is niet verplicht.

Bij de functie aanroepen kan voor deze parameters gebruik worden gemaakt van het define “NIET” wat in dit geval staat voor “0.0/ReadOnly”. Op deze wijze kan het totaal aantal extra parameters fors worden beperkt.

```
/*      Ondergrens V.A. groen bij alternatieve realisatie                  */
/*      -------------------------------------------------                  */
/*      Indien NIET dan wordt de ondergrens V.A. groen bij alternatieve    */
/*      realisatie gelijk verondersteld aan de ingestelde vastgroentijd    */

        ALTVA02,          8.0,        /* signaalgroep 02                  */
        ALTVA05,          8.0,        /* signaalgroep 05                  */
        ALTVA08,          8.0,        /* signaalgroep 08                  */
        ALTVA11,          8.0,        /* signaalgroep 11                  */

/*      Percentage V.A. groen voor kappen door OV ingreep                  */
/*      -------------------------------------------------                  */
/*      De maximum groenduur van de primaire realisatie wordt bepaald      */
/*      door de parameters START##_x en EINDE##_x. Het ondermaximum        */
/*      voor afbreken is instelbaar als een percentage van deze            */
/*      maximum groenduur.                                                 */
/*        0%: Alleen de vastgroentijd is gegarandeerd                      */
/*      100%: De primaire realisatie mag niet worden ingekort              */
/*      Indien NIET dan wordt als instelling “0” verondersteld.            */

        KAPOV02,         80.0,        /* signaalgroep 02                  */
        KAPOV05,         80.0,        /* signaalgroep 05                  */
        KAPOV08,         80.0,        /* signaalgroep 08                  */
        KAPOV11,         80.0,        /* signaalgroep 11                  */

/*      Instellingen tbv prioriteit openbaar vervoer -> signaalgroep 02    */
/*      ---------------------------------------------------------------    */
        DTTS02,          20.0, /* default tijd      tot  stopstreeppassage */
        DVGR02,           3.0, /* default groentijd voor stopstreeppassage */
        UITB02,          30.0, /* duur uitmeldbewaking                    */
        PVEU02,           3.0, /* max.tijd startgroen voor einde uitstel  */

/*      Instellingen tbv prioriteit openbaar vervoer -> signaalgroep 05    */
/*      ---------------------------------------------------------------    */
        DTTS05,          20.0, /* default tijd      tot  stopstreeppassage */
        DVGR05,           3.0, /* default groentijd voor stopstreeppassage */
        UITB05,          30.0, /* duur uitmeldbewaking                    */

/*      Afbreek methode door openbaar vervoer stappenraster 1              */
/*      -----------------------------------------------------              */
/*      0.0: Inkorten    groenfase nooit toegestaan                        */
/*      1.0: Uitstellen  groenfase toegestaan                              */
/*      2.0: Afbreken    groenfase toegestaan                              */
/*      4.0: Doorsnijden groenfase toegestaan                              */

/*      De afbreek methode is instelbaar als optelsom per signaalgroep     */
/*      vb: 1.0: Alleen het uitstellen van de groenfase is toegestaan      */
/*          7.0: Alle afbreek methoden zijn toegestaan                     */
/*      Indien NIET dan wordt als afbreekmethode "7" verondersteld.        */

        ABMOV02_1,        7.0,        /* signaalgroep 02                  */
        ABMOV05_1,        7.0,        /* signaalgroep 05                  */
        ABMOV08_1,        7.0,        /* signaalgroep 08                  */
        ABMOV11_1,        7.0,        /* signaalgroep 11                  */

/*      Toestemming prioriteit OV in "blokkade"gebied                      */
/*      ---------------------------------------------                      */
/*      0.0: Prioriteit tijdens "blokkade"gebied is nooit  toegestaan      */
/*      1.0: Prioriteit tijdens rood  voor "einde uitstel" toegestaan      */
/*      2.0: Prioriteit tijdens groen na   "afkap moment"  toegestaan      */

/*      De toestemmingsmethode is instelbaar als optelsom per signaalgroep */
/*      vb: 2.0: Alleen aanhouden van het groen na het "afkap moment" is   */
/*               toegestaan                                                */
/*          3.0: Prioriteit is altijd toegestaan                           */
/*      Indien NIET dan wordt als toestemmingsmethode "0" verondersteld.   */

/*      Indien er geen uitstel- en afkapmoment is gedefinieerd is          */
/*      prioriteit buiten het primaire gebied per definitie toegestaan.    */

        TPMOV02_1,        1.0,        /* signaalgroep 02 - stappenraster 1 */
        TPMOV02_2,        0.0,        /* signaalgroep 02 - stappenraster 2 */
        TPMOV02_3,        0.0,        /* signaalgroep 02 - stappenraster 3 */

        STOP};
```

8.3. Opbouw crapcod.c

Vanaf versie 5.0 is het mogelijk om de CRSV module te combineren met een RWS-C "blokken" regeling. Als van deze combinatie gebruik wordt gemaakt dient het volgende define statement te worden opgenomen:

#define CRSV_EN_BLOK
LET OP: Dit define moet geplaatst worden voor de include van crsv.c.

De CRSV functies worden aangeroepen vanuit crapcod.c. De volgorde van deze functie aanroepen ligt vast. De broncode van de CRSV module is vanaf versie 6.01 vrij beschikbaar en dient in crapcod.c te worden opgenomen door middel van het volgend include statement:

#include "CRSV.C"

Init bij inschakelen:

```
void init_bij_inschakelen(void)
{
/* Initialisatie CRSV module */
/* ------------------------- */
   proc_init_crsv(FIXATIE,PROGKEUZE,EG_CRSV_VERSIE);
```

“NIET” ipv “FIXATIE” schakelt het fixeren uit.
De CRSV module schrijft het versie nummer in EGGPARM EG_CRSV_VERSIE.

```
/* Definitie wachttijdvoorspellers */
/* ------------------------------- */
/* #define WTV_BUS */
WTV_SG[SG24] = TRUE;
```

In dit voorbeeld is SG24 gekoppeld aan een wachttijdvoorspeller. De wachttijdvoorspeller is niet voorzien van een BUS sjabloon. (#define WTV_BUS is als commentaar opgenomen)

```
/* Definitie voetgangersfietskoppeling */
/* ----------------------------------- */
   proc_def_vtg_fts(SG32,SG21,SG22,MR2122,MR2221);
```

Indien slechts één fietsrichting aanwezig dan als argument 3 t/m 5 “NIET” meegeven.

```
/* Definitie progressieve voetgangersoversteek */
/* ------------------------------------------- */
   proc_def_vtg_bb(SG32,SG92,T3292,GKGA3292,SG21,SG22,MR2122,MR2221,GKGP3292);
```

In dit voorbeeld is SG32 de “binnenste” en SG92 de “buitenste” signaalgroep.
Indien slechts één fietsrichting aanwezig dan als argument 7 t/m 9 “NIET” meegeven.

```
/* Definitie gescheiden voetgangersoversteek */
/* ---------------------------------------- */
   proc_def_vtg_go(SG31,SG32,SG31_1,SG32_2,T3132,T3231,GKGA3132,
                   SG21,SG81,MR2181,TB2181,TE2181,HNL2181,
                   SG22,SG82,MR2282,TB2282,TE2282,HNL2282,MR2122,MR2221,GKGP3132
                   NIET,NIET,NIET  ,NIET  ,NIET  ,NIET   ,NIET  );
```

In dit voorbeeld zijn SG81 en SG82 de volgrichtingen van SG21 en SG22
Indien volgrichting SG81 niet aanwezig dan als argument 9 t/m 13 “NIET” meegeven.
Indien volgrichting SG82 niet aanwezig dan als argument 15 t/m 19 “NIET” meegeven.
Indien slecht één fietsrichting aanwezig dan als argument 14 t/m 21 “NIET” meegeven.

```
/* Definitie seriele koppeling */
/* --------------------------- */
   set_HF(HKI_0262)
   proc_def_hki(SG02,SG62,TV0262,TB0262,TE0262,TM0262,SG02_1,NIET,NIET,
                     SG72,TV6272,TB6272,TE6272,TM6272,SG62_1,NIET,NIET,HKI_0262);
```

In dit voorbeeld is SG62 de volgrichting van SG02 én SG72 de volgrichting van SG62.
De koppeling wordt aktief bij het opstarten van het regelprogramma.

```
/* Definitie OV beïnvloeding */
/* ------------------------- */
   proc_def_ov(SG42,DTTS42,DVGR42,UITB42,NIET);
}
```

In dit voorbeeld heeft SG42 geen einde uitstel moment.

Voorwaarden per seconde:

```
void voorwaarden_per_seconde(void)
{
   proc_crsv_per_seconde();   /* bepaal signaalgroep klokken */
}
```

Overige voorwaarden:

```
void overige_voorwaarden(void)
{
/* BEPAAL SCHAKELBARE SERIELE KOPPELINGEN */
/* ====================================== */
   HF(HKI_0262) = SCH(S_HKI_0262);
```

In dit voorbeeld is de seriële koppeling instelbaar door middel van een software schakelaar.

```
/* BEPAAL PROGRAMMA-KEUZE EN STATUS STAPPENRASTER SLAAF */
/* ==================================================== */
   proc_crsv_progkeuze_slaaf(3,DEFPRG,T_OMSC,H_KPIN1,H_KPIN2,H_KPIN3,H_KPIN4);

   if (AKTPRG == 1) proc_crsv_parm(CTYD_1,INSC_1,OMSC_1,SYNC_1);
   if (AKTPRG == 2) proc_crsv_parm(CTYD_2,INSC_2,OMSC_2,SYNC_2);
   if (AKTPRG == 3) proc_crsv_parm(CTYD_3,INSC_3,OMSC_3,SYNC_3);

   proc_crsv_stappenraster(H_KPIN5);
```

In dit voorbeeld zijn 3 stappenraster gedefinieerd.
H_KPIN1 t/M H_KPIN5 zijn hulpfuncties mbt de 5 inkomende koppelsignalen.

```
/* BEPAAL PROGRAMMA-KEUZE EN STATUS STAPPENRASTER COMPLEXCOORDINATOR */
/* ================================================================ */
   TL(VAS) = 1;        /* vastlegging programma keuze */

   proc_crsv_progkeuze_master(3,DEFPRG,T_OMSC,H_KPUIT1,H_KPUIT2,H_KPUIT3,H_KPUIT4,VAS);

   if (AKTPRG == 1) proc_crsv_parm(CTYD_1,INSC_1,OMSC_1,SYNC_1);
   if (AKTPRG == 2) proc_crsv_parm(CTYD_2,INSC_2,OMSC_2,SYNC_2);
   if (AKTPRG == 3) proc_crsv_parm(CTYD_3,INSC_3,OMSC_3,SYNC_3);

   proc_crsv_master_raster(H_KPUIT5,T_SYNC);
```

In dit voorbeeld zijn 3 stappenraster gedefinieerd en wordt altijd stappenraster 1 geselecteerd.
H_KPUIT1 t/m H_KPUIT5 zijn hulpfuncties mbt de 5 uitgaande koppelsignalen.

```
/* BUFFER IN- EN UITMELDINGEN OPENBAAR VERVOER */
/* ========================================== */
   if (begin_D(SG42_2)) {
        _tts = NIET; _vgr = NIET; _vtg = 0; _pri = 2;
        proc_ov_inm(SG42);
   }
   if (begin_D(SG42_1)) {
        _vtg = 0;
        proc_ov_uit(SG42);
   }
```

In dit voorbeeld is SG42_2 de inmelddetector en SG42_1 de uitmelddetector voor SG42.
Dmv _tts wordt de (vrije) tijd tot stopstreep passage meegegeven. (NIET = kies default)
Dmv _vgr wordt de tijd meegegeven nodig om het verkeer voor de stopstreep te laten afrijden.
(NIET = kies default)
Dmv _vtg kan het voertuignummer worden meegegeven. (0 = voertuignummer onbekend)
Dmv _pri wordt de prioriteit voor het voertuig meegegeven.
(0 = geen prioriteit, 1 = alleen aanhouden eigen groenfase, 2 = prioriteitsrealisatie)

```
/* BEPAAL AKTUELE PRIMAIRE GEBIEDEN */
/* ================================ */
   if (AKTPRG == 1) {
       proc_akt_crsv(SG02,CPRIM02,UITST02_1,START02_1,WACHT02_1,EINDE02_1,AFKAP02_1,
                     ABMOV02_1,TPMOV02_1);
       proc_akt_crsv(SG05,CPRIM05,UITST05_1,START05_1,WACHT05_1,EINDE05_1,AFKAP05_1,
                     ABMOV05_1,NIET);
       proc_akt_crsv(SG08,CPRIM08,UITST08_1,START08_1,WACHT08_1,EINDE08_1,AFKAP08_1,
                     ABMOV08_1,NIET);
       proc_akt_crsv(SG11,CPRIM11,UITST11_1,START11_1,WACHT11_1,EINDE11_1,AFKAP11_1,
                     ABMOV11_1,NIET);
   }
   if (AKTPRG == 2) {
        etc.
   }


/* BEPAAL ONDERGRENS V.A. GROEN BIJ ALTERNATIEVE EN CONFLICTERENDE PRIORITEITSREALISATIE */
/* ================================================================================== */
   proc_kapm_crsv(SG02,ALTVA02,KAPOV02);
   proc_kapm_crsv(SG05,ALTVA05,KAPOV05);
   proc_kapm_crsv(SG08,ALTVA08,KAPOV08);
   proc_kapm_crsv(SG11,ALTVA11,KAPOV11);

/* BEPAAL OPENHOUDEN PRIMAIR GEBIED DOOR APPLICATIE */
/* ================================================ */
   proc_open_prim(SG02,H_OPEN_PRIM_SG02);
```

Zolang hulpfunctie H_OPEN_PRIM_SG02 waar is kan het aanvraaggebied van SG02 niet vervroegd worden afgesloten.

```
/* CRSV-MODULE SIGNAALGROEPSTURING DEEL 1 */
/* ===================================== */
/* Corr.ALTM[] - KAPM[] agv gekoppelde oversteken: proc_corr_kapm(); */
/* Bepaal tijd tot einde groen,             TEG[]: proc_bteg_crsv(); */
/* Bepaal tijd tot alternatieve realisatie, TTR[]: proc_bttr_crsv(); */
/* Bepaal tijd tot primaire     realisatie, TTP[]: proc_bttp_crsv(); */
/* Bepaal beurt    prioriteits  realisatie       : proc_prio_crsv(); */
/* Bepaal beurt    primaire     realisatie       : proc_prim_crsv(); */
/* Bepaal toestemming alt.realisatie CRSV-module : proc_altt_crsv(); */

   CRSV_SG_main_1();


/* BEPAAL HERSTART NALOOP SERIELE FIETSKOPPELINGEN */
/* =============================================== */
   HF(HNL2181) = TRUE;
   HF(HNL2282) = D(SG22_1);
```

In dit voorbeeld wordt de naloop van SG21 naar SG81 gedurende de gehele groenfase van SG21 (her)start. De naloop van SG22 naar SG82 alleen indien detector D(SG22_1) bezet is.

```
/* BEPAAL BLOKKEER ALTERNATIEVE REALISATIE DOOR APPLICATIE */
/* ======================================================= */
   proc_altt_appl(SG02,H_BLOK_AR_SG02);
```

Zolang hulpfunctie H_BLOK_AR_SG02 kan SG02 niet alternatief realiseren.

```
/* CRSV-MODULE SIGNAALGROEPSTURING DEEL 2 */
/* ===================================== */
/* Bepaal beurt alternatieve realisatie          : proc_altr_crsv();     */
/* Afwikkeling fiets.voetgangersoversteek        : proc_vtg_fts_crsv();  */
/* Afwikkeling progressieve voetgangersoversteek : proc_bb_vtg_crsv();   */
/* Afwikkeling gescheiden   voetgangersoversteek : proc_go_vtg_crsv();   */
/* Afwikkeling Harde Koppeling Intern            : proc_hki_crsv();      */
/* Bepaal fictieve wachtstand.groen.aanvraag     : proc_wsgr_fca_crsv(); */

   CRSV_SG_main_2();

/* AFWIKKELING WACHTSTAND GROEN */
/* ============================ */
   proc_wsgr_crsv(SG02,WGR02);
   proc_wsgr_crsv(SG08,WGR08);

/* AFWIKKELING MEEVERLENGGROEN IN GEVAL VAN 1 KRUISPUNT BINNEN DE REGELAAR */
/* ======================================================================= */
   if (hf_mvg()) {
       proc_mvgr_crsv(SG02,MVG02);
       proc_mvgr_crsv(SG05,MVG05);
       proc_mvgr_crsv(SG08,MVG08);
   }

/* AFWIKKELING MEEVERLENGGROEN IN GEVAL VAN MEERDERE KRUISPUNTEN BINNEN DE REGELAAR */
/* ================================================================================ */
   if (hf_mvg_krp(SG02,SG11)) {
       proc_mvgr_crsv(SG02,MVG02);
       proc_mvgr_crsv(SG05,MVG05);
       proc_mvgr_crsv(SG08,MVG08);
   }
   if (hf_mvg_krp(SG62,SG71)) {
       proc_mvgr_crsv(SG62,MVG62);
       proc_mvgr_crsv(SG65,MVG65);
       proc_mvgr_crsv(SG68,MVG68);
       proc_mvgr_crsv(SG71,MVG71);
   }
```

In geval van meerdere kruispunten binnen de regelaar kan op deze wijze voorkomen worden dat signaalgroepen van een kruispunt meeverlengen met signaalgroepen van een ander kruispunt. Voorwaarde is dat in crapdef.h alle signaalgroepen van een kruispunt opeenvolgend zijn gedefinieerd.

In dit voorbeeld is SG02 de 1^{e} signaalgroep en SG08 de laatste signaalgroep van kruispunt 1. SG62 is de 1^{e} signaalgroep en SG71 de laatste signaalgroep van kruispunt 2.

```
/* KOPPELING CRSV-MODULE -> CRS */
/* ============================ */
   proc_crsv_to_crs();
}
```

Binnen de functie proc_crsv_to_crs() worden de in paragraaf 8.1. beschreven signaalgroep opties en de WSGR_toewijzing() bestuurd.

9. Historisch overzicht CRSV

1 augustus 1998: **Versie 1.0**
Oplevering eerste versie.

1 oktober 2000: **Versie 2.0**
Toevoegingen:

- Fietsvoetgangerskoppeling zoals omschreven in paragraaf 6.1.
- Seriële koppeling zoals omschreven in paragraaf 6.4.
- Hulpfunctie PRM_APPL_SG() zoals omschreven in paragraaf 5.6.
- Toevoeging SCH(GKGnm) aan de voetgangerskoppelingen.

Wijzigingen:

- Einde uitstel refereert aan het vroegste begin_GROEN_moment ipv het vroegste begin_ROG_moment. (zie paragraaf 3.1.1.)

7 augustus 2001: **Versie 2.1**
Toevoegingen:

- EPARM(PROGKEUZE) zoals omschreven in paragraaf 3.1.7.
- Toevoeging van TTPG[] tbv koppeling met de wachttijdvoorspeller.

Wijzigingen:

- De definities van EPARM(UITSTn_x), EPARM(WACHTn_x) en EPARM(AFKAPn_x) zijn niet meer verplicht.
  Bij de functie aanroepen kan gebruik worden gemaakt van het define “NIET”.

24 november 2002: **Versie 2.2**
Wijzigingen:

- De programma keuze wordt bepaald door 4 ipv 3 koppelsignalen.
- Uitbreiding EPARM(PROGKEUZE). Toegevoegd is de keuze 1.1; lokaal met versneld doorstappen stappenraster.

10 januari 2004: **Versie 2.2a**
Bug opgelost:

- Bug opgelost in de bepaling van TTPG[].

26 februari 2006: **Versie 2.2b**
Bug opgelost:

- Bug opgelost in de afwikkeling van de seriële koppeling.
- Bepaling TTPG[] verplaatst om rekentijd te besparen.

1 juli 2008: **Versie 2.3**

Wijzigingen:

- Voetgangersfietskoppelingen uitgebreid tot maximaal twee fietsrichtingen per koppeling. Bij de gescheiden voetgangersoversteek kunnen deze twee fietsrichtingen bovendien ieder weer een volgrichting hebben.
- Bij de gescheiden voetgangersoversteek kan de voedende richting al “inlopen” tijdens het RVG() van de volgrichting.
  Alleen indien er geen gekoppelde fietsrichting is gedefinieerd.

5 mei 2009: **Versie 2.3a**

Bug opgelost:

- Bug opgelost in het doorschrijven van fictieve conflicten bij de gescheiden voetgangersoversteek.

11 oktober 2009: **Versie 3.0**

Toevoeging:

- Beïnvloeding door openbaar vervoer zoals omschreven in hoofdstuk 7.

Wijziging:

- Documentatie volledig (her)schreven.

1 maart 2011: **Versie 4.0**

Bug opgelost:

- Bug opgelost in de gescheiden voetgangersoversteek. Een drukknop die op blijft staan kon een vastloper veroorzaken.
- Start aanvraag kon opkomen nadat de aanvraag al aktief was geworden. Dit kon een vastloper veroorzaken.

Wijzigingen:

- Seriële koppelingen schakelbaar en uitgebreid tot maximaal drie signaalgroepen.
- Toekennen van alternatieve realisaties aan een seriële koppeling gewijzigd zodat inrijden onder het groen van het laatste conflict is toegestaan.
- Signaalgroepen worden bij het omschakelen van een stappenraster alleen versneld gerealiseerd indien er onvoldoende alternatieve realisatieruimte is.

Toevoeging:

- Koppeling met (31 led)wachttijdvoorspeller zoals omschreven in paragraaf 5.8.

7 augustus 2011: **Versie 4.0a**

Bug opgelost:

- Bug opgelost in hiaatmeting. Een afgebroken hiaatmeting kon pas worden herstart nadat de betreffende detector afgevallen was.
- Afbreken conflictrichtingen bij inrijden voedende richting van een seriële koppeling verbeterd. (conflictrichtingen konden te vroeg worden afgebroken)

1 mei 2012: **Versie 4.1**

Wijziging:

- Bij programmakeuze “Stappenraster LOKAAL met versneld doorstappen” wordt versneld doorgeschakeld vanaf het moment dat een signaalgroep een aanvraag voor groen heeft. (dit was vanaf het moment dat de signaalgroep groen toonde buiten het primaire gebied)

8 april 2013: **Versie 4.1a**

Bug opgelost:

- Vooruitrealisaties van toevoerende richtingen bij een interne harde koppeling werden geblokkeerd als de afvoerende richting geen aanvraag had.

21 december 2013: **Versie 4.2**

Bug opgelost:

- Afhandeling vooruitrealisatie op einde primair gecorrigeerd.

Wijziging:

- Buffer versie CRSV module in EGGPARM EG_CRSV_VERSIE.

16 februari 2014: **Versie 5.0**

Wijziging:

- CRSV module geschikt gemaakt voor geïntegreerde versie C-regelaar.

1 juni 2015: **Versie 6.0**

Wijziging:

- Gescheiden voetgangersoversteek uitgebreid tot maximaal drie signaalgroepen.

Bug opgelost:

- Indien een voedende richting van een harde koppeling intern alternatief realiseerde binnen zijn eigen primaire gebied dan kon dit leiden tot een overschrijding van de maximale voorstarttijd en het niet realiseren van de nalooptijden.

11 oktober 2015: **Versie 6.01**

Wijziging:

- CRSV module geschikt gemaakt voor zowel de geïntergreerde als de niet geïntergreerde versie van de C-regelaar.
- Bugfix harde koppeling intern. (te vroeg inkomen van de voedende richitng)