DINAMO

Interface specificatie 3.0

Auteur: Leon J.A. van Perlo
Versie: 3.0
Datum: 9 juli 2007

INHOUD

Release Beheer

Dit document is van toepassing op release:

- RM-H: Release 3.0
- TM-H: Release 5.0
- TM-CC: Release 1.0
- OM32: Release 1.1, 1.2

Leeswijzer

Dit document bestaat uit 5 gedeelten

Deel 1 behandelt de communicatie tussen computer en Dinamo en is feitelijk alleen interessant voor degenen die zelf besturingssoftware willen schrijven.

Deel 2 behandelt de commandostructuur van Dinamo. Deze is van belang voor degenen die zelf besturingssoftware schrijven, maar kan ook nuttig zijn indien je bepaalde initialisatiecommando's naar Dinamo wilt zenden. Dit kan via een separaat hulpprogramma, door middel van Koploper of met eigen software.

Deel 3 behandelt de configuratie van Dinamo Event Actions ten behoeve van Virtuele Uitgangen en configuratie van schakelaars/bezetmelders.

Deel 4 beschrijft hoe Dinamo op een eenvoudige manier geïnitialiseerd en geconfigureerd kan worden.

Voorwoord

Dinamo is oorspronkelijk ontwikkeld als geavanceerd blokbesturingssysteem voor 2-rail modelspoorbanen in elke willekeurige schaal met uitsluitend analoge locs. Anno 2006 bleken een aantal zaken aan heroverweging toe te zijn. Zo begon een aantal gebruikers tegen de fysieke limiet van 64 blokken aan te lopen. Deze limiet was met het oude protocol 2.x niet oplosbaar daar er geen ruimte was om de extra adresbits onder te brengen. Ten tweede levert een aantal fabrikanten in toenemende mate rollend materieel in uitsluitend digitale uitvoering en gedigitaliseerde locs rijden niet op een pulsbreedte gestuurd bloksysteem.

Daarnaast vindt de modelspoorbaan steeds vaker een uitbreiding met rijdende modelauto's, bijvoorbeeld op basis van het Faller Car System. De besturingsmogelijkheden van dit systeem zijn echter zeer beperkt. Ten behoeven van Railz Miniworld in Rotterdam is een methode ontwikkeld om auto's op een geavanceerde manier kunnen besturen. De auto's worden uitgerust met een decoder die via een draadloos zend/ontvangsysteem commando's dient te ontvangen. De commando's en signalen voor deze decoders kunnen met de Dinamo hardware worden opgewekt.

Bij "Car Control" wordt het blokgestuurde principe van Dinamo losgelaten. De 8 uitgangen op een TM-CC worden parallel gebruikt om meerdere zendlussen apart te kunnen aansturen.

Om bovengenoemde redenen kent Dinamo vanaf versie 3.0 een aantal uitbreidingen, met name:

- De mogelijkheid om analoge en DCC locs op 1 spoorbaan door elkaar te laten rijden, mits analoge en DCC locs niet tegelijk in 1 blok aanwezig zijn. Uiteraard kunnen wel meerdere DCC locs in 1 blok aanwezig zijn en onafhankelijk van elkaar bestuurd worden. Ook kunnen meerdere analoge locs in 1 blok aanwezig zijn, maar die kunnen dan niet onafhankelijk van elkaar bestuurd worden.
- Een capaciteitsuitbreiding tot 128 blokken per systeem, met de optie deze verder te vergroten tot 256 + uitbreiding van andere zaken die hier logischerwijs uit volgen.
- De mogelijkheid auto's op een geavanceerde manier te besturen. Hiertoe dienen de TM-H controllers te worden vervangen door TM-CC controllers, op basis van dezelfde hardware, maar met andere software. Op dit moment is het slechts mogelijk om een Dinamo systeem te gebruiken voor auto's OF treinen, niet tegelijk. In de toekomst kan deze flexibiliteit waarschijnlijk wel worden toegevoegd.

De ze handleiding beschrijft het protocol dat gebruikt wordt vanaf versie 3.0 voor de besturing van Dinamo. Protocol 3.0 wijkt fors af van protocol 2.x en een redelijke compatibiliteit was niet haalbaar.

Wijzigingen ten opzichte van 2.x

Adresbyte:
- Bit 3 staat op 1 als indicatie dat het protocol versie 3 betreft
- Er is een extra "message size" bit bijgekomen (bit 2, zodat de berichtgrootte nu reikt van 0 tot 7

Noot: bits 2 en 3 waren in protocol versie 2 "adresbits". Deze zijn feitelijk nooit echt gebruikt in het protocol tussen PC en RM51. Met ingang van protocol 3.0 vervallen de adresbits dus volledig om ruimte te maken voor andere zaken. De benaming "adresbyte" is dus ook niet meer correct, maar we zullen hem omwille van 'herkenbaarheid' voorlopig handhaven.

Als de berichtgrootte groter dan 0 is, is het eerste databyte een commando. Deze commando's zijn in protocol 3 volledig verschillend van versie 2.
Het protocol is (nog steeds) een 7-bits protocol. Het 8^{e} (meest significante) bit (bit 7) wordt steeds gebruikt om aan te geven of het een adresbyte of databyte betreft.

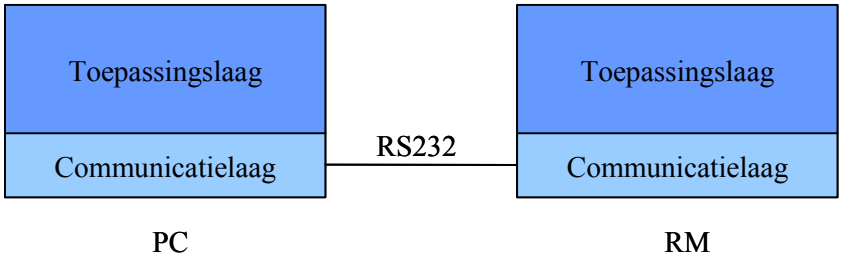
Vervallen

Met ingang van 3.0 zijn ook een aantal mogelijkheden vervallen die nauwelijks nog nut hadden of in de nieuwe opzet niet bruikbaar of niet meer mogelijk zijn. De belangrijkste:

- Het optioneel uitvoeren van Switch-event-actions op basis van de richtingsinstelling van een blok is (in elk geval voorlopig) komen te vervallen.
- Een aantal event-actions is komen te vervallen.
- Bij 2.x werden bij gelinkte blokken alle door de RM51 voor de gelinkte blokken gegenereerde commando's gemeld aan de besturingsPC. Bij versie 3.0 gebeurt dat niet meer. Alleen events, dus gebeurtenissen die op basis van tijd of een gebeurtenis optreden worden gemeld. Dit betreft:
  - Analoge versnellingsstappen gegenereerd door de RM-H
  - Het beeindigen van de bekrachtiging van een magneetartikel
  - Het sluiten of openen van een schakelaar of bezetmelder
  - Kortsluiting of het opheffen daarvan
  - Antwoorden op een status-request van de PC

1 Communicatiestructuur

Communicatie tussen de RM-H en besturende PC (een willekeurige computer met RS232 poort) is het best te beschrijven met een communicatiemodel van 2 lagen. De onderste laag noemen we de communicatielaag, de bovenste laag noemen we de toepassingslaag.



1.1 De communicatielaag

De communicatielaag zorgt voor het in stand houden van een foutloze en betrouwbare verbinding. Communicatie vindt plaats via een asynchrone RS232 verbinding op 19.200bps met 8 bits en odd parity (poort instellingen). Het communicatieprotocol is van een Master-Slave type. De Master (PC) is verantwoordelijk voor het in stand houden van de communicatie met een continue stroom berichten. De Slave (RM) werkt hieraan mee door het terugsturen van een bericht na ontvangst van een bericht van de PC. Een bericht op het niveau van de communicatielaag noemen we hierna ***datagram***. De term ***bericht*** gebruiken we voor de op de toepassingslaag daadwerkelijk betekenisvolle inhoud (payload) van een datagram.

De PC initieert de communicatie door het verzenden van een datagram. Indien de RM dit foutloos ontvangt, stuurt de RM een datagram terug. Zodra de PC dit foutloos heeft ontvangen stuurt deze een nieuw datagram, etc. Indien de PC niet binnen een vastgestelde tijd (200 ms is een goede praktische waarde) een foutloos datagram terugkrijgt, stuurt deze het vorige datagram opnieuw, wacht op antwoord, etc.

Als een ontvangen datagram (aan om het even welke zijde) een fout bevat wordt dit volkomen genegeerd en geacht niet te zijn ontvangen.

Indien de RM gedurende 2 seconden geen enkel (foutloos) datagram ontvangt stopt het om veiligheidsredenen onmiddellijk alle treinen en gaat in “Fault” mode. De RM neemt hiermee aan dat ofwel de PC is gecrashed, de PC applicatie is vastgelopen of de communicatie onherstelbaar is verstoord.

Elk datagram heeft de volgende structuur: een adres-byte, optioneel 1 tot 7 data-bytes en een checksum byte.

P	7	6	5	4	3	2	1	0	← Bitnummer
P	0	T	F	H	1	X2	X1	X0	Address Byte
P	1	D6	D5	D4	D3	D2	D1	D0	Data Byte(s)
P	1	CS6	CS5	CS4	CS3	CS2	CS1	CS0	Checksum Byte

P = Parity (odd, oneven)

De totale lengte van een datagram is dus minimaal 2 bytes en maximaal 9.

Voor de communicatielaag is de bericht inhoud van de datagrammen totaal niet relevant. Indien de communicatielaag een datagram ontvangt met bericht-inhoud (geen NULL-datagram). Levert het het betreffende bericht af aan de toepassingslaag. Evenzo controleert de communicatielaag alvorens een datagram weg te sturen of er in de toepassingslaag een bericht klaarstaat ter verzending. Indien dit zo is vult het de data bytes van het datagram met deze informatie en verstuurt het. Indien er geen informatie te verzenden is stuurt de communicatielaag een NULL datagram.

1.2 Adres-byte

X0, X1 en X2 geven het aantal data bytes weer in het datagram. Een leeg datagram (NULL datagram) heeft dus X0=0, X1=0 en X2=0 en bestaat in totaal uit 2 bytes: Het adresbyte en checksum byte.

Bit 3 staat vast op 1. Dit is de indicatie dat het protocol versie 3.x is. Bij versie 2.x waren bits 2 en 3 altijd 0.

Bit 4 is een HOLD bit. Als de RM-H in korte tijd veel commando's krijgt die hij niet allemaal kan verwerken of bufferen (hetgeen b.v. kan gebeuren als je echt heel veel wissels ineens wilt omzetten, zie verder) zet de RM het HOLD bit. De bedoeling hiervan is de PC te informeren dat de buffers vol beginnen te raken en te verzoeken verdere opdrachten even op te schorten. De PC moet dan (liefst) alleen nog NULL datagrammen sturen. In principe mag je ook bij H=1 berichten (=datagrammen met inhoud) sturen en de RM zal deze ook proberen te verwerken, echter je loopt dan het risico dat er een foutconditie ontstaat (zie verder).

Omgekeerd kan ook de PC in de berichten van PC naar RM-H de HOLD flag zetten. De RM-H zal dan geen informatie naar de PC sturen maar alleen NULL datagrammen. De data wordt in de RM-H gebufferd. Let op dat het buffer in de RM-H beperkt is tot 128 berichten en dat het daarom wenselijk is dergelijke situaties tot een minimum (duur) te beperken. Toch kan het gebruik van het HOLD bit in deze vorm handig zijn, bijvoorbeeld bij het configureren van Event Actions. Dit wordt vaak uitgevoerd door een apart programmagedeelte en tijdens dit configureren wil je wellicht even niet gestoord worden door gebeurtenissen vanuit Dinamo, omdat je op dat moment niet in staat bent die berichten te verwerken. Het geval wil dat het HOLD bit niet van toepassing is op "Event Action Configuration Responses". Heb je dus het HOLD bit naar de RM-H gezet, dan kun je slechts "EAC-Responses" of NULL-datagrammen terugkrijgen.

Bit 5 is een FAULT bit en geeft aan dat de RM in foutmodus staat in welk geval alle treinen gestopt zijn. Een foutstatus kan ontstaan in de RM-zelf of in een van de aangesloten TM-H subsystemen. Een fout in de RM-H treedt op als de RM meer dan 2 seconden geen datagram van de PC heeft ontvangen, of er een interne buffer-overflow optreedt (bijvoorbeeld als de PC toch messages blijft sturen terwijl H=1). Uit veiligheidsoverwegingen gaat dan de noodstop er op. Een fout kan ook ontstaan in een van de aangesloten TM-H subsystemen. Indien de foutstatus actief is kan de PC de bron van de fout achterhalen door het opvragen van foutinformatie (zie verder).

Als de PC van mening is dat de kust veilig is dient deze de foutconditie te resetten (bericht). Er rijden geen treinen zolang F=1. Verder worden alle commando's normaal uitgevoerd (voor zover mogelijk) en alle events gerapporteerd aan de PC. Als je dus wilt dat de treinen na het resetten van een fout langzaam optrekken zul je de snelheden voor het geven van het Reset Fault commando eerst op 0 moeten zetten.

Een FAULT treedt meestal niet zomaar op, tenzij je een extreem trage PC hebt die niet in staat is voldoende NULL datagrammen te sturen. Het verdient daarom aanbeveling een

opgetreden fout niet zo maar volautomatisch te resetten, maar liefst eerst na te gaan wat de oorzaak is, eventueel door de gebruiker om een bevestiging te vragen.

Omgekeerd reageert de RM-H ook op de FAULT flag in de berichten van PC naar Dinamo. Als deze gezet is worden alle treinen gestopt Zodra een datagram wordt ontvangen met F=0 nemen de treinen de oorspronkelijke snelheid weer aan (zonder vertraagd optrekken). Het F-bit is een extreem snelle en efficiënte manier om vanuit de PC een noodstop te genereren zonder ook maar 1 inhoudelijk bericht te hoeven sturen.
Bit 6 is een “Toggle” bit. Bij elk nieuw datagram vanuit de PC wordt dit bit door de PC omgezet, bij hertransmissie houdt dit bit z’n waarde. Stel dat de RM wel een goed datagram ontvangt en een response stuurt die door de PC niet (correct) wordt ontvangen. De PC pleegt dan een hertransmissie. Aan het T-bit kan de RM zien dat dit een hertransmissie is en hij de eventuele inhoud van het datagram niet nogmaals moet afleveren aan de toepassingslaag, maar alleen een nieuwe response (= hertransmissie van het vorige bericht) moet sturen. T-response = T-host.

Bit 7 = 0 voor een address byte. De ontvanger kan hieraan altijd de start van en nieuw datagram herkennen ten behoeve van hersynchronisatie na een transmissiefout.

1.3 Data-bytes

Omvat de daadwerkelijke nuttige inhoud (bericht) van een datagram af te leveren aan de toepassingslaag. Aantal data bytes = 0 (NULL-Datagram) tot 7.

Bit 7 = 1 (Databyte indicatie)

1.4 Checksum-byte

Het Checksum-byte is het one’s complement van de som van alle voorgaande bytes in het datagram, inclusief het address byte, zodat, als eindresultaat, de som van alle bytes in het datagram nul is. Bit 7 wordt hierbij genegeerd.
Voor verzending berekenen door alle voorgaand bytes van het datagram op te tellen (zonder carry), dan alle bits omzetten, 1 optellen en bit 7 op 1 forceren.
Ontvangen datagram controleren door alle bytes (inclusief checksum) op te tellen en controleren of het resultaat modulo 128 gelijk aan 0 is.

1.5 De toepassingslaag

De toepassingslaag bekommert zich niet om de betrouwbaarheid van de communicatie. De communicatielaag verzekert immers een nagenoeg 100% betrouwbare verbinding. Zoals de communicatielaag zich niet interesseert voor de berichtinhoud van de datagrammen, interesseert de toepassingslaag zich niet voor datagrammen. Datagrammen zijn op toepassingslaag ook niet zichtbaar. De communicatielaag levert ontvangen berichten af aan de toepassingslaag en ontvangt te verzenden berichten van de toepassingslaag.

De interne communicatie tussen toepassingslaag en communicatielaag (aan PC zijde) hangt uiteraard af van de implementatie door de maker van de software. In elk geval moeten ontvangen berichten door de communicatielaag kunnen worden afgeleverd bij de toepassingslaag en moeten berichten van de toepassinglaag kunnen worden aangeboden aan de communicatielaag. De berichtlengte moet bij de aflevering van een bericht worden gecommuniceerd aangezien dit in vele gevallen mede bepalend is voor de betekenis van het bericht.

2 RM-H Berichten Definitie

Berichten zijn te onderscheiden in commando's (command), antwoorden (response) en gebeurtenissen (event).

- Een commando (C) is een opdracht die de PC aan de RM controller geeft
- Een event (E) is een 'spontane' gebeurtenis die de RM controller zelf meldt aan de PC
- Een response (R) is een antwoord van de RM op een commando van de PC

Onderstaande tabel bevat een overzicht van de in protocol 3.0 bestaande berichten. Na de tabel volgt een uitgebreide toelichting

Lengte (data bytes)	Byte 1 (bin) Commando	Byte 2 (bin) Sub-commando	Beschrijving	Versie (update)	Type
>2	0000000		Subsysteem commando		C
2	0000001	0000000	Reset fout		C
3	0000001	0000001	Stel HFI niveau in		C
2	0000001	0000010	Vraag versie nummer		C/R
3	0000001	0000011	Vraag systeem info		C/R
3	0000001	0000100	Stel "flash period" in		C
3	0000001	0000101	Stel opties in		C
2	0000001	0000110	Subsystem debugging uit		C
2	0000001	0000111	Subsystem debugging aan		C
3	0000001	0001001	Subsystem retransmission		E
3	0000001	0100000	Configureer Event Action		C
3	0000001	100SSSS	Ontkoppel schakelaar		C
4	0000001	11ESSSS	Koppel schakelaar aan Event Action		C
3	0000101		Verwijder aansturing van auto		C
5	0000101		Stuur One-Off pakket voor auto		C
4/5	0000110		Zet snelheid van auto		C
4/5	0000111		Zet functies van auto		C
2/3	00010CC		Digitale uitgang		C
2	00011CU		Virtuele uitgang		C
2/3	0010CMM		Magneetartikel uitgang		C
4	0011MMM		OM32 uitgang		C
3/4	010000B		Analoge functie / snelheid		C
3/4	010001B		Analoge snelheid met polariteit		C
4/5	010100B		DCC functie / snelheid		C
4/5	010101B		DCC snelheid met polariteit		C
2	01100CB		Alarm event		E
2	01101CB		Alarm status aanvraag / response		C/R
3	011100B		Unlink		C
4	011101B		Copy / Link		C
3	011110B		Kickstart (analog)		C
3	011111B		Blokbesturing		C
2	10CSSSS		Schakelaar event		E
2	11CSSSS		Schakelaar status aanvraag/response		C/R

In onderstaande toelichting:

(….) staat voor een byte in het bericht
[(….)] staat voor een optioneel byte in het bericht
{(….)} staat voor een of meerdere bytes in het bericht

>> Betekent: Dit is een commando
<< Betekent: Dit is een event of response
<> Betekent: Dit kan zowel een commando als een response zijn

2.1 Systeemcommando's

>> Subsystem Command (0000000) (TTYYYYY) {(xxxxxxx)}

Hiermee kan via de RM-H een rechtstreeks bericht worden gestuurd naar een subsysteem, zonder dat dit bericht eerst door de RM-H geïnterpreteerd wordt. Het volgende databyte geeft aan naar welk subsysteem het bericht moet, de daarop volgende bytes zijn het daadwerkelijk doorgestuurde bericht. Door de RM-H worden derhalve de eerste 2 bytes van dit commando gestript en het restant doorgestuurd aan het aangegeven subsysteem (max 5 bytes dus).

TTYYYYY is de adressering van het betreffende subsysteem
YYYYY is het indexnummer van het subsysteem (adres) 0..31
TT is het type subsysteem:

- 00 = Niet gedefinieerd
- 01 = TM-H
- 10 = PM32
- 11 = OM32

Voor de mogelijkheden en betekenis van subsysteem commando's wordt verwezen naar de desbetreffende handleiding/specificatie

<> System Control (0000001) (CCCCCCC) {(xxxxxxx)}

- (0000001) (0000000) Reset Fault
- (0000001) (0000001) (HFI Level) Set HFI Level (0..15)
- (0000001) (0000010) Version Info Request
- (0000001) (0000011) (Type) System Info Request
- (0000001) (0000100) (Period) Set Flash Period
- (0000001) (0000101) (Options) Set Options
- (0000001) (0000110) Subsystem Debugging OFF
- (0000001) (0000111) Subsystem Debugging ON
- (0000001) (0001001) (TTYYYYY) Subsystem Retransmission
- (0000001) (0100000) (Bytes) Event Action Configuration
- (0000001) (100SSSS) (sssssss) Clear Switch Event Action
- (0000001) (11ESSSS) (sssssss) (EventAction) Set Switch Event Action

Reset Fault

Schoont de foutstatus van de RM-H en die van de aangesloten TM-H subsystemen

Set HFI Level

Dinamo biedt in blokken die in analoge modus werken de mogelijkheid permanente verlichting van rollend materieel te realiseren via een (HF) wisselspanning.
Set HFI-Level stelt het niveau van de HFI verlichting in in 16 stappen. 0 = uit, 15 = maximaal vermogen. HFI-level is identiek voor alle blokken en alle aangesloten TM-H subsystemen. Per (analoog) blok kan gekozen worden of HFI aan of uit staat (zie verder).

Version Info Request

Geeft als antwoord de versie van de RM-H:

<< (0000001) (0000010) (0MMMmmm) (0sssbbb)

MMM = Major Release
mmm = Minor Release
sss = subrelease
bbb = bugfix
Versie 3.12a (bugfix a op versie 3.12) zou derhalve geven:

<< (0000001) (0000010) (0 011 001) (0 010 001)

System Info Request

Hiermee is het mogelijk extra informatie te krijgen m.b.t. de toestand van het systeem.

Info type 0 geeft in een bericht de stand van de dipswitches op de RM-H

<< (0000001) (0000011) (0 0 0 0 s4 s3 s2 s1)

sx = 0: switch on, sx = 1: switch off
De betekenis van de switches hangt af van de firmware release. Om hiermee iets zinvols te doen moet je dus eigenlijk eerst het release nummer opvragen.

Info type 1 geeft in een bericht een indicatie van de interne piekbelasting van het systeem.

<< (0000001) (0000011) (0010000) (<cap>)

Het derde byte is 0x10 (decimaal 16). Het vierde byte geeft de ongebruikte verwerkingscapaciteit van het systeem aan tijdens de maximale piekbelasting die de RM te verwerken heeft gehad. Op een 22 MHz systeem is deze maximaal 96. Indien de gerapporteerde restcapaciteit 0 is, is ooit de maximale verwerkingscapaciteit van het systeem bereikt. De RM-H kan prima omgaan met overbelasting (gaat hierdoor niet down), maar als de restcapaciteit in de buurt van 0 komt kan het betekenen dat er (herstelbare) communicatiefouten ontstaan en (met name) cycli overgeslagen worden. Het effect van dit laatste is bv dat knipperende uitgangen langzamer gaan knipperen.

Info type 2 geeft aan welke TM-H subsystemen actief zijn.

<< (0000001) (0000011) (010 t3 t2 t1 t0) (010 t7 t6 t5 t4) (010 t11 t10 t9 t8) (010 t15 t14 t13 t12)

Info type 3 geeft aan welke TM-H subsystemen een foutstatus rapporteren. Indien de RM-H een FAULT geeft aan de PC, kan dit ofwel een interne fout zijn ofwel een fout van een van de aangesloten TM-H's

<< (0000001) (0000011) (011 f3 f2 f1 f0) (011 f7 f6 f5 f4) (011 f11 f10 f9 f8) (011 f15 f14 f13 f12)

Info type 6 geeft aan welke PM32 subsystemen actief zijn.

<< (0000001) (0000011) (110 p3 p2 p1 p0) (110 p7 p6 p5 p4)

Info type 127 reset de piekbelasting indicator (geeft geen response)

Set Flash Period

Past de standaard Flash-on periode voor de RM-H digitale uitgangen aan. De ingestelde Flash-periode is alleen van toepassing op nieuwe 'flitsers'. Eenmaal lopende 'flitsers' worden door dit commando niet beïnvloed.

Set Options

Stel "speciale" opties in. Het "Options" byte is bit-georienteerd. Momenteel is gedefinieerd:

Bit	Uit (0)	Aan(1)	Vanaf versie
0	OM32 enkelvoudige transmissie	OM32 dubbele transmissie	
1	-	Subsysteem hertransmissie rapport	
2			
3			
4			
5			
6			

Subsystem Debugging

Zorgt ervoor dat alle naar subsystemen gestuurde opdrachten ook worden teruggestuurd naar de besturings-PC. Dit gebeurt zonder enige filtering. Daardoor kunnen deze commando's identiek zijn aan normale responses die een andere betekenis hebben.
Subsystem Debugging is alleen bedoeld voor "factory testing" (eigenlijk alleen mij zelf dus) en mag bij normaal bedrijf NOOIT worden ingeschakeld, tenzij je verrassende zaken wilt meemaken.

Subsystem Retransmission

Als het betreffende "options"-bit gezet is (zie "Set Options") stuurt de RM-H dit bericht naar de PC indien een hertransmissie naar een subsysteem moet worden uitgevoerd. Deze optie is bedoeld om de stabiliteit van het communicatienetwerk (tussen RM-H en de aangesloten subsystemen TM-H en PM32) te kunnen testen.
De rapportage werkt pas nadat de 'subsystem discovery' is afgerond (d.w.z. als de aangesloten subsystemen zijn gevonden en alleen indien de RM-H zelf NIET in foutmodus staat.
De definitie van het subsysteem-nummer is gelijk aan die bij "send subsystem command"

Event Action Configuration

Dinamo kent 256 zogenaamde "Event Actions". Dat zijn acties die Dinamo zelf kan nemen aan de hand van bepaalde gebeurtenissen. Deze gebeurtenissen kunnen zijn het activeren of de-activeren van een schakelaar(/bezetmelder) of het ontvangen van een commando

vanuit de besturings-PC. Event Actions moeten, als je hiervan gebruik maakt, geconfigureerd worden.

Event Action Configuration bestaat uit een bericht, gevolgd door een Event Action pakket. Het pakket zelf wijkt af van het normale Dinamo berichtenformaat en heeft een “payload” van 8 bits per byte. De werking qua overdracht is dus dat de PC eerst een EAC bericht stuurt naar Dinamo waarin de komst van het pakket wordt aangekondigd en het aantal bytes dat dit pakket bevat. Vervolgens wordt het pakket gestuurd (8 bits per byte, oneven polariteit),

In antwoord op het ontvangen pakket geeft de RM51 een Event Action Config Response. Dit is een aan een EAC identiek bericht, waarbij (Bytes) vervangen is door de volgende waarde:

  0 = OK
  1 = Content Error (het ontvangen pakket voldoet niet aan de voorschriften)
  2 = Checksum Error
255 = Parity Error (tenminste één van de bytes had een parity error)

Set/Clear Switch Event Action

Met “Set” wordt een schakelaar/bezetmelder (0..2047) gekoppeld aan een van de 256 Event Actions. “E” in het subcommand is het meest significante bit van het Event Action nummer
Met “Clear” wordt de schakelaar/bezetmelder ontkoppeld van het Event Action

2.2 Besturing van auto’s

Auto’s worden bestuurd met Model-Car-Control paketten. Dinamo genereert MCC pakketten voor de aanwezige auto’s. Een MCC basispakket bevat in principe alle informatie voor de betreffende auto.

- Snelheid
- Optrek –en remkarakteristieken
- Licht, remlichten, knipperlichten
- Functies F1..F4

Voor verdere details wordt verwezen naar de MCC specificaties.

Op dit moment (TM-CC versie 1.00) kunnen per systeem 64 MCC pakketten worden gegenereerd voor 64 (actieve) auto’s per systeem. Als Dinamo een instructiepakket ontvangt (snelheid of functies) voor een auto die nog niet wordt aangestuurd wordt dit pakket automatisch toegevoegd.

Als een auto fysiek wordt verwijderd uit het (deel) systeem dient de aansturing van deze auto door de besturingsPC te worden verwijderd.

Indien een instructiepakket voor een ‘nieuwe’ auto wordt ontvangen en het systeem is vol, dan verwijdert Dinamo zelf de aansturing van de auto die het langst geen nieuwe instructie heeft ontvangen. Dinamo stuurt in dit geval een “verwijder” bericht naar de besturingsPC.

Naast de normale ‘operationele’ pakketten kunnen programmeer –of configuratiepakketten worden gestuurd. MCC decoders kennen, vergelijkbaar met DCC decoders, Configuratie-Variabelen waarmee eigenschappen van de decoder kunnen worden ingesteld.

Verwijder (adressering van) auto

<> Verwijder (0000101) (00AAAAA) (aaaaaaa)

- AAAAAaaaaaaa = decoder adres (1..4095)

Verwijder pakketten voor decoder uit de berichtenstroom

One-Off pakketten t.b.v. configuratie van MCC decoders

One-Off wil zeggen dat het pakket slechts 1 maal gestuurd wordt en niet in de cyclische berichtenstroom wordt toegevoegd. Het 'One-Off' commando is identiek aan het 'Verwijder' commando. Dit is opzettelijk zo gedaan aangezien een 'One-Off' commando tevens het pakket voor de betreffende aansturing verwijdert om interferentie met programmeerpakketten te voorkomen.

>> Program-Start (0000101) (01AAAAA) (aaaaaaa) (000 00BB) (SSS SSSS)

- AAAAAaaaaaaa = decoder adres (1..4095)
- BB = CV-bank die geadresseerd gaat worden:
  - 00 = 0..63
  - 01 = 64..127
  - 10 = 128..191
  - 11 = 192..255
- SSS SSSS = Het aantal pakketten dat zal volgen

Het Program-Start pakket zet de auto volledig stil en zet MCC decoder in programmeermode. Het aantal data-pakketten dat volgt dient EXACT het aantal te zijn dat met de parameter SSS SSSS wordt opgegeven. Het maximaal aantal kan per MCC decoder variëren.

>> Program-Data (0000101) (10AAAAA) (aaaaaaa) (VVV DDDD) (vvv dddd)

- AAAAAaaaaaaa = decoder adres (1..4095)
- VVV vvv = De CV die geadresseerd wordt. Het volledige CV adres wordt gevormd door de CV-Bank uit het Program-Start commando en de VVV vvv bits
- DDDD dddd = De data die in de CV geschreven wordt

De programmeergegevens van de Program-Data pakketten worden door Dinamo verzonden en worden door de MCC decoder tijdelijk in RAM opgeslagen.

>> Program-Finish (0000101) (11AAAAA) (aaaaaaa) (000 0000) (000 0000)

- AAAAAaaaaaaa = decoder adres (1..4095)
- Het 4^e en 5^e byte zijn 0. Ze moeten worden meegstuurd, maar worden door Dinamo vervangen door checksum-data die automatisch gegenereerd wordt.

Indien de juiste sequentie van Program-Start, Program-Data en Program-Finish door de MCC decoder wordt ontvangen zal deze de ontvangen gegevens naar het Flash-geheugen schrijven.

Snelheid

>> Snelheid (0000110) (00AAAAA) (aaaaaaa) (00DSSSS) [(0000 XXX)]

- AAAAAaaaaaaa = decoder adres (1..4095)
- SSSS = snelheid. 0 = stop, 1 = minimum snelheid, 15 = maximaal
- D = richting, 1 = vooruit, 0 = achteruit, momenteel niet gebruikt, dus altijd 1
- XXX = acceleratie-parameter. Hiermee kan aangegeven worden hoe snel de ingestelde snelheid bereikt moet worden 0 = onmiddellijk, 7 = maximale vertraging

Als de auto (adres) nog niet werd aangestuurd wordt dit toegevoegd. Het laatste byte is optioneel. Als dit niet wordt meegestuurd blijft de laatst ingestelde waarde gehandhaafd. Is er geen vorige waarde, dan wordt XXX = 000

Functies

>> Functies (0000111) (00AAAAA) (aaaaaaa) (000 R L B H) [(000 F4 F3 F2 F1)]

- AAAAAaaaaaaa = decoder adres (1..4095)
- H = Licht (Headlights)
- B = Remlicht (Brake) [1]
- L = richting Links
- R = richting Rechts
- F1..F4 = Aanvullende functies

[1] Noot: Remlichten kunnen ook door de auto zelfstandig worden aangestuurd. In dat geval hoeft het remlicht door de besturingsPC alleen te worden aangestuurd als de remlichten aan moeten zijn terwijl de auto stil staat.

Als de auto (adres) nog niet werd aangestuurd wordt dit toegevoegd. Het laatste byte is optioneel. Als dit niet wordt meegestuurd blijft de laatst ingestelde waarde gehandhaafd. Is er geen vorige waarde, dan wordt dit F1..F4 = 0

2.3 Besturing van uitgangen

>> Uitgang (00010CC) (uuuuuuu) [(parameter)]

Het betreft hier de aansturing van de digitale uitgangen die rechtstreeks door de RM-H (RM51) (parallel) worden bestuurd. Dit zijn er evanals in protocol 2.x maximaal 128.

- uuuuuuu = 0..127 is de uitgang die aangestuurd wordt
- CC = het commando voor de betreffende uitgang. Dit hangt mede af van de extra parameter die wel of niet kan worden meegestuurd.

  Zonder extra parameters zijn de opdrachten:
  - 00 = Uit
  - 01 = Inverteer
  - 10 = Aan
  - 11 = Ongedefinieerd

  Met extra parameter zijn de opdrachten:
  - 00 = Knipper met (parameter) = halve periode
  - 01 = Flash: Periodiek aan gedurende een vooraf ingestelde periode, uit gedurende (parameter). De “aan” periode kan met het “Set Flash Period” commando worden ingesteld.
  - 10 = Puls: aan gedurende (parameter), daarna uit
  - 11 = Randomize: Elk (parameter) interval willekeurig aan/uit

De tijdseenheid is 1/60 sec
De commando’s komen qua functie overeen met de commando’s uit protocol 2.x

>> Virtuele Uitgang (00011CU) (uuuuuuu)

Het betreft hier de uitvoering van wat in protocol 2.x “Switch Event Actions” heette. De werking is vergelijkbaar met dien verstande dat m.i.v. 3.0 een “SEA” niet meer hard gekoppeld is aan een schakelaar/bezetmelder.
Er zijn 256 mogelijke Virtuele Uitgangen per systeem, dus identiek aan versie 2.x met 2 mogelijke condities per V.U. (aan of uit)

- Uuuuuuuu = 0..255 is de V.U. die wordt aangestuurd
- C = de stand van de V.U. die wordt aangestuurd (0/1)

<> Magneetartikel (0010CMM) (mmmmmmm) [(tijd)]

Zet een puls op een magneetartikel uitgang (wisselspoel, spoelen van vormseinen, etc)

- MMmmmmmmm = 0..511 is het magneetartikel (spoelnummer) dat wordt aangestuurd
- C = de stand waarin die spoel gezet wordt (0 = rechtdoor, 1 = afbuigend)
- Tijd (optioneel) is de tijdsperiode gedurende welke de spoel bekrachtigd wordt in eenheden van 1/60 sec. Default = 200ms

Als de spoel is omgezet stuurt Dinamo een respons (identiek bericht, maar zonder (tijd)) ter bevestiging.
LET OP: Dit is geen garantie dat de wissel ook fysiek is omgezet, alleen dat de puls is gegeven en is beëindigd.

>> OM32 Uitgang (0011MMM) (mmuuuuu) (commando) (parameter)

Hiermee wordt een OM32 commando (commando) met parameter (parameter) voor uitgang (uuuuu) aan OM32 module (MMMmm) gegeven. De noodzakelijke 2-traps benadering van versie 2.0 komt hiermee te vervallen.
LET OP: een alternatief om OM32 commando's te sturen is via Subsystem Command. Dit kost 1 byte meer, maar hier wordt het modulenummer en uitgangsnummer niet gemengd. Verder is de werking identiek.

2.4 Besturing van snelheid en functies van locomotieven

Dinamo versie 3.0 ondersteunt 2 typen locomotieven. Analoge locs zonder decoder en digitale locs met DCC decoder.

Analoge locs worden aangestuurd met pulsbreedtemodulatie. Hierbij kan tevens permante verlichting worden gegenereerd door middel van een hoogfrequent wisselspanning (HFI). De polariteit van het blok bepaalt de rijrichting en, bij HFI, de zijde van de loc waaraan het licht brandt. Richtingsafhankelijke verlichting werkt alleen met een extra hulpschakeling in de loc.

DCC locs worden bestuurd met DCC paketten. Dinamo genereert per blok DCC pakketten voor de in dat blok aanwezige locs. Momenteel worden 4 typen paketten ondersteund:

- Snelheid
- Licht en functies F1..F4
- Functies F5..F8
- Functies F9..F12

De rijrichting wordt bepaald door het richting-bit in het DCC snelheidspakket (en dus niet door de polariteit van het blok).
Voor verdere details wordt verwezen naar de DCC specificaties. Op dit moment (versie 3.00) kunnen per blok 10 paketten worden gegenereerd.

In een Dinamo systeem kunnen analoge en DCC locs door elkaar worden gebruikt, echter geen combinatie van verschillende types tegelijkertijd in één blok. Een blok staat in analoge of DCC modus en kan door middel van commando's worden omgeschakeld.

Tevens kan een blok volledig stroomloos worden gemaakt door het softwarematig uitschakelen van het vermogen (Power-off).

Bij Dinamo 2.x kon de (analoge) snelheid van –64..+63 lopen. Bij DCC wordt de rijrichting bepaald ten opzichte van de loc, bij analoog bedrijf wordt dit bepaald ten opzichte van het blok. Er zijn vanaf heden 2 variabelen die de richting kunnen beinvloeden, het "P" bit (polariteit) en het "R" bit (DCC). Om niet de verwarring te krijgen op welke variabele het negatief zijn van de snelheid betrekking heeft is besloten de snelheid zelf voortaan als absolute waarde te hanteren en de 2 richtingsbits los hiervan te behandelen. De absolute snelheid (analoog) loopt van 0..63. DCC snelheid loopt van 0..28. Op dit moment wordt (bij DCC) uitsluitend 28 stappen modus ondersteund.

Analoge snelheid/functies

(0100xxB) (bbbbbbb) (xxxxxxx) [(xxxxxxx)]

- Bbbbbbbb = bloknummer

Als het blok Bbbbbbbb in DCC modus staat wordt het blok omgezet naar analoog. De DCC pakketinfo voor het betreffende blok wordt dan gewist. Initieel is de snelheid 0 en HFI verlichting uit. De polariteit blijft bij omschakeling behouden, behalve bij die functies waar de polariteit expliciet wordt meegegeven.

LET OP: Indien het blok is uitgeschakeld (power-off) zorgen deze commando's voor een automatische inschakeling (auto-power-on)

>> Snelheid analoog (010000B) (bbbbbbb) (1SSSSSS) [(AAAAAAA)]

- SSSSSS (0..63) bepaalt de absolute snelheid
- AAAAAAA (optioneel) = 0..127 bepaalt de gesimuleerde massatraagheid. Hoe groter deze waarde hoe langer de tijd (in eenheden van1/60 sec) tussen 2 snelheidsstappen. AAAAAAA = 0 of het helemaal weglaten van deze parameter leidt tot een onmiddellijke instelling van de nieuwe snelheid

Bij een versnelling (massasimulatie) genereert Dinamo bij elke stap een snelheids"event" om aan de besturingsPC aan te geven wat de daadwerkelijk ingestelde snelheid is. Dit is steeds een snelheid analoog met polariteit (zie verder)

>> Functie analoog (010000B) (bbbbbbb) (00L0000)

Met deze opdracht kan de HFI verlichting voor blok Bbbbbbbb worden aan of uit gezet. Het format van het tweede en derde byte is voor deze functie identiek aan die van het overeenkomstige DCC commando, zodat je vanuit de programmatuur analoge en digitale locs desgewenst op vergelijkbare wijze kunt behandelen.

- L = 1: licht aan, L = 0: licht uit

LET OP: Om het licht aan de gewenste zijde van de loc te laten branden moet je HFI aan zetten en de juiste Polariteit instellen (werkt alleen bij een geplaatste lichtwisselschakeling)

Het aan en uitzetten van HFI verlichting kan ook via blokbesturing (zie verder)

<> Snelheid analoog met polariteit (010001B) (bbbbbbb) (PSSSSSS) [(AAAAAAA)]

- SSSSSS (0..63) bepaalt de absolute snelheid
- P bepaalt de polariteit, 1 = positief, 0 = negatief
- AAAAAAA (optioneel) = 0..127 bepaalt de gesimuleerde massatraagheid. Hoe groter deze waarde hoe langer de tijd tussen 2 snelheidsstappen. AAAAAAA = 0 of het helemaal weglaten van deze parameter leidt tot een onmiddellijke instelling van de nieuwe snelheid

Bij een versnelling (massasimulatie) genereert Dinamo bij elke stap een snelheids"event" om aan de besturingsPC aan te geven wat de daadwerkelijk ingestelde snelheid is.

DCC snelheid/functies

>> (0101xxB) (bbbbbbb) (xxxxxxx) (ddddddd) [(DDDDDDD)]

- Bbbbbbbb = bloknummer

Als het blok Bbbbbbbb in analoge modus staat wordt het omgezet naar DCC. De analoge snelheid van het blok wordt in dat geval 0 gemaakt en HFI wordt uit gezet. Initieel is alle DCC pakketinfo van het blok gewist. De polariteit wordt behouden, behalve bij die functies waar de polariteit expliciet wordt meegegeven.

LET OP: Indien het blok is uitgeschakeld (power-off) zorgen deze commando's voor een automatische inschakeling (auto-power-on)

>> DCC Snelheid (010100B) (bbbbbbb) (1RSSSSS) (ddddddd) [(DDDDDDD)]

- SSSSS (0..28) bepaalt de absolute snelheid (29 t/m 31 worden teruggezet naar 28)
- R bepaalt de rijrichting van de loc, 1 = vooruit, 0 = achteruit
- ddddddd (0..127) is het decoder basisadres waarvoor de snelheid wordt gegenereerd. Als [(DDDDDDD)] niet aanwezig is betreft het een 7 bits decoder adres. Als [(DDDDDDD)] wel aanwezig is (ook als [(DDDDDDD)] = 0 !!) betreft het een 14 bit decoderadres, waarbij DDDDDDD de 7 meest dignificante bits zijn en ddddddd de 7 minst significatie bits. Het maximum decoder adres is 10239 (decimaal)
  ddddddd = 0 is een "broadcast" (decodernummer 0). Werkt niet voor alle decoders.
  LET OP: De 2 meest significante bits van het hoge orde 14-bit decoder-adresbyte, die bij DCC dan altijd 11 zijn worden NIET meegestuurd, maar door Dinamo zelf gegenereerd. Je stuurt dus alleen de 14 daadwerkelijke adresbits, Dinamo mapt dit op 2 bytes van 8 bits met de hoogste 2 bits 11

>> DCC Functiegroep 1, 2a, 2b (010100B) (bbbbbbb) (0XXFFFF) (ddddddd) (DDDDDDD)

Hiermee stuur je functies Licht en F1 t/m F12 aan van een DCC decoder

- XX = 00: FFFF besturen F1 tm F4 (bit 0 is F1) Licht = uit
- XX = 01: FFFF besturen F1 tm F4 (bit 0 is F1) Licht = aan
- XX = 10: FFFF besturen F9 tm F12 (bit 0 is F9)
- XX = 11: FFFF besturen F5 tm F8 (bit 0 is F5)

DDDDDDDddddddd bepalen het decodernummer zoals bij Snelheid DCC

>> DCC snelheid met polariteit (010101B) (bbbbbbb) (PRSSSSS) (ddddddd) [(DDDDDDD)]

- SSSSS (0..28) bepaalt de absolute snelheid (29 t/m 31 worden teruggezet naar 28)
- R bepaalt de rijrichting van de loc, 1 = vooruit, 0 = achteruit
- P bepaalt de polariteit, 1 = positief, 0 = negatief
- DDDDDDDddddddd bepalen het decodernummer zoals bij Snelheid DCC

Let op: De polariteit kan (uiteraard) alleen per blok worden ingesteld, niet per loc/decoder in hetzelfde blok. Het blok krijgt de polariteit van het laatst ontvangen commando met polariteitsinstelling.

2.5 Besturing van overige blokgeorienteerde functies

<< Alarm event (01100CB) (bbbbbbb)

Dit is een EVENT gegenereerd door Dinamo
- C=1: blok Bbbbbbbb heeft kortsluiting
- C=0: kortsluiting van blok Bbbbbbbb is opgeheven

<> Alarm Request/Response (01101CB) (bbbbbbb)

Hiermee kun je opvragen of een blok Bbbbbbbb in alarmstatus staat. Bij het commando is C niet relevant, bij het antwoord hierop van Dinamo geeft Dinamo in C de status weer.

>> Link (011101B) (bbbbbbb) (0000IPS) (sssssss)

- Bbbbbbbb = het blok dat gekoppeld wordt (het “bestemmingsblok”)
- Ssssssss = het blok waaraan gekoppeld wordt (het “source”blok)
- P = Permanent: P=0: maak eenmalige kopie, P=1: maak een link
- I = Inverteer polariteit

Het “link” commando maakt een exacte kopie van alle instellingen van het bron-blok naar het bestemmingsblok. De enige uitzondering hierop is een eventuele kickstart instelling. Een exacte kopie is belangrijk in het geval een trein van één blok naar het volgende rijdt. De elektrische aansturing van beide blokken moet in dat geval 100% identiek en synchroon zijn, anders ontstaat er kortsluiting.
In principe kun je 2 blokken ook een gelijke instelling geven door ze beide identieke opdrachten te geven. Bij versie 2.x was dat nog vrij gemakkelijk, maar indien je ook DCC loks laat rijden (bij versie 3.0) wordt dat lastiger. Beide blokken moeten dan namelijk niet alleen dezelfde DCC pakketten genereren, maar dat ook synchroon en in dezelfde volgorde doen. De enige garantie daarvoor is als je beide blokken eerst volledig schoont en vervolgens in dezelfde volgorde dezelfde instellingen geeft of wanneer je gebruik maakt van het copy / link commando. Dat laatste heeft verreweg de voorkeur, omdat het veel minder opdrachten vergt en de kans op fouten verkleint.

Indien het “P” bit niet gezet is (P=0), maakt link een eenmalige kopie, verder niets. Indien het “P” bit wel gezet is (P=1) maakt “link” en permanente link. Dit betekent dat hierna elk commando naar het bron-blok automatisch wordt doorgegeven aan het bestemmings-blok. Een gelinkt blok kan op zijn beurt weer worden doorgelinkt, zodat een kettiing ontstaat. Ook kan een reeds gelinkte ketting in zijn geheel worden gekoppeld aan een nieuw bron-blok. Pas met dit laatste enigszins op: indien de ketting erg lang is en het nieuwe bron-blok veel aktieve DCC pakketten bevat ontstaat hierdoor een grote informatie-“broadcast” in het Dinamo systeem. In (zeer) extreme gevallen kan dit leiden tot een buffer-overflow en een foutsituatie. Koppel daarom nieuwe blokken liever aan het eind van een bestaande ketting i.p.v. een ketting aan een nieuw bron-blok.

Een bron-blok kan tegelijkertijd maar één bestemming hebben en een bestemmingsblok maar één bron-blok. Met een ketting zou het gevaar kunnen bestaat dat een kringverwijzing ontstaat. Indien een “link”commando hiertoe zou leiden wordt het commando niet uitgevoerd.

Een link kan worden verboken door:
1. Het “unlink” commando;
2. het sturen van een willekeurige opdracht naar het bestemmingsblok

Indien het “I” bit gezet is (I=1) wordt de polariteit geïnverteerd. D.w.z. dat de elektrische signalen van beide gelinkte blokken tegengesteld zijn. Dit is in bepaalde gevallen noodzakelijk bij blokken die bv onderdeel zijn van een keerlus.

>> Unlink (011100B) (bbbbbbb) (0000UZ0)

- Bbbbbbbb = bloknummer
- U = 1: Unlink-Up, U = 0: Unlink Down
- Z = Maak blokken die ontkoppeld worden 0 (schoon alle DCC data en zet snelheid op 0)

Een link tussen 2 blokken kan worden verbroken door een willekeurig commando te sturen naar het zogenaamde bestemmingsblok. Nadat een link op een dergelijke manier verbroken is blijft de bestaande informatie in beide blokken (of beide deel-kettingen) in stand en gaat een gescheiden leven leiden.

Een andere manier om links te verbreken is door middel van “Unlink”. Door middel van een optie-vlag kun je een unlink-up of een unlink-down uitvoeren.

Unlink-down verbreekt het betreffende blok van alle daaraan gekoppelde blokken en die blokken onderling. De hele ketting vanaf het betreffende blok wordt dus afgebroken. Uiteraard wordt het betreffende blok ook losgekoppeld van een eventueel “bovenliggend” blok (er wordt immers een “willekeurig” command naar dat blok gestuurd). De optionele Z-vlag zorgt er voor dat alle onkoppelde blokken ook meteen worden geschoond van alle data en stroomloos worden gemaakt.

Unlink-up verbreekt een blok van alle bovenliggende blokken en tevens die bovenliggende blokken onderling. Het betreffende blok wordt dus de “top” van een eventueel nog resterende ketting. Met de optionele Z-vlag kunnen alle ontkoppelde blokken worden geschoond en stroomloos worden gemaakt. Let op: Dit geldt in dit geval NIET voor het betreffende blok zelf!

>> Kickstart (011110B) (bbbbbbb) (waarde)

KickStart is primair bedoeld om motoren met een hoge ankerkleef die zeer moeilijk op gang komen een beter rijgedrag te geven. Voor de toepassing van KickStart is een aparte notitie geschreven.
Zet KickStart van blok Bbbbbbbb op (waarde). Indien waarde = 0: Kickstart uit.

Kickstart werkt alleen als het blok reeds in analoge modus staat.

Let op: Kickstart wordt doorgegeven aan gekoppelde blokken **(dit wijkt dus af van de implementatie in release 2.x)** en het geven van een kickstart aan een volg-blok leidt ook tot het verbreken van de ketting, net zoals dat met andere commando’s gebeurt. Echter een eenmaal aan een blok of ketting gegeven kickstart instelling wordt NIET doorgegeven als hieraan vervolgens weer extra blokken worden gekoppeld.

>> Blokbesturing (011111B) (bbbbbbb) (ADXPPUU)

Hiermee kan een blok worden in of uitgeschakeld, kan de gewenste modus van blok worden ingesteld en kunnen gegevens worden gewist. De bitpatronen ADX, PP en UU kunnen onafhankelijk van elkaar of in combinatie worden gebruikt.

- ADX = 000: Geen actie
- ADX = 001: Laat blok in huidige (A/D) modus staan, wis alle DCC pakketinfo, zet snelheid op 0 en HFI uit.
- ADX = 10x: Zet blok bbb analoog. Wis eventuele opgeslagen DCC paketten.
  - X = 0: HFI (licht) uit
  - X = 1: HFI aan
- ADX = 01x: Zet blok bbb in DCC modus. Zet een eventuele analoge snelheid op 0 en HFI uit.
  - X = 1: Wis alle DCC pakketinfo

- PP = 00 Geen actie
- PP = 10 Zet Pol(Bbbbbbbb) = 0 (negatief)
- PP = 11 Zet Pol(Bbbbbbbb) = 1 (positief)

- UU = 00 Geen actie
- UU = 10 Schakel blok Bbbbbbbb uit
- UU = 11 Schakel blok Bbbbbbbb in

2.6 Besturing van schakelaars (bezetmelders)

<< Switch (10CSSSS) (sssssss)

Dit is een EVENT gegenereerd door Dinamo
- C=1: schakelaar SSSSssssssss is geactiveerd
- C=0: schakelaar SSSSssssssss is gedeactiveerd

<> Switch Request/Response (11CSSSS) (sssssss)

Aangezien de RM zelf alleen switch events genereert weet je de status van een schakelaar pas als er tenminste 1 event is opgetreden. Bij schakelaars is dat normaliter niet bezwaarlijk, maar voor bezetspoordetectoren (die door de RM ook gewoon als schakelaar worden gezien) wel. Door middel van een Switch Status Request commando kun je daarom expliciet de status van een specifieke schakelaar (of detector) opvragen.
In het commando is C niet relevant. In het antwoord geeft Dinamo de actuele status: actief (C=1) of niet-actief (C=0)

3 DINAMO Configuratie

Dinamo kent een aantal interessante configuratiemogelijkheden, die het systeem een aantal functies autonoom kunnen doen uitvoeren. Indien extreem doorgevoerd kan Dinamo zelfs zelfstandig functioneren als besturingseenheid voor bv een pendelbaan, doch normaliter worden deze autonome functies gebruikt om de besturingscomputer (host) te ontlasten van oninteressante taken of om functies met een real-time karakter te implementeren.

Dinamo beschikt over de mogelijkheid “Event Actions” (EA) op te slaan. Elke EA kent 2 toestanden, “ON” en “OFF”. Elke EA/Toestand combinatie kan maximaal 2 instructies bevatten. Er zijn in totaal 512 EA’s, 256 voor de toestand “ON” en 256 voor de toestand “OFF”

Een EA kan worden gegenereerd vanuit 2 bronnen: De PC of een schakelaar/bezetmelder. Het genereren vanuit de PC gebeurt via de instructie “Virtuele Uitgang”. Het geven van het commando “virtuele uitgang 23 aan” leidt dus tot het uitvoeren van de opgeslagen instructies van “Event Action 23 ON”.
Om EA’s te genereren vanuit schakelaars/bezetmelder dient die schakelaar te worden gekoppeld aan het betreffende EA. Indien dat is gebeurd zal bij het openen van de betreffende schakelaar het EA-OFF worden uitgevoerd en bij het sluiten EA-ON.

3.1 Event Action Flags

Het uitvoeren van de diverse instructies wordt gecontroleerd door “flags”. Deze flags geven per EA, per toestand, per bron en per instructie aan of deze wel of niet moet worden uitgevoerd. Deze “flags” kunnen dynamisch worden aangepast door Event Actions, zodat het optreden van een bepaalde gebeurtenis de uit te voeren acties als gevolg van een andere gebeurtenis kan beïnvloeden en er op die wijze onderlinge afhankelijkheden kunnen worden gerealiseerd.

Elk van de 512 Event Actions heeft een flag-byte met de volgende betekenis:

Bit 0 = EA Instructie 1 uitvoeren bij optreden Schakelaar Event
Bit 1 = EA Instructie 2 uitvoeren bij optreden Schakelaar Event
Bit 2 = EA Instructie 1 uitvoeren bij ontvangen van een opdracht Virtuele Uitgang
Bit 3 = EA Instructie 2 uitvoeren bij ontvangen van een opdracht Virtuele Uitgang
Bit 4 = Doorgifte Schakelaar Event aan PC blokkeren

3.2 Configureren van Event Actions

Event Actions kunnen worden geconfigureerd door het sturen van een Event Action pakket (zie hoofdstuk 2). Het pakket is 12 bytes lang en heeft de volgende inhoud:

<toestand> <EA-nr> <EAF> <EA1(4 bytes)> <EA2(4 bytes)> <csum>

<toestand>	0 of 1. 0 Correspondeert met OFF, 1 correspondeert met ON.
<EA-nr>	is het nummer van de te configureren switch.
<EAF>	Event Action Flags (zie par 3.1)
<EA1>	Event Action Instructie 1 bestaande uit 4 bytes
<EA2>	Event Action Instructie 2 bestaande uit 4 bytes
<csum>	controlewaarde, heeft een zodanige waarde dat de som modulo 256 van alle bytes in het pakket gelijk is aan nul.

3.3 Event Action Instructies

Elke EA-Instructie bestaat uit 4 bytes. In onderstaande tabel vind je een overzicht. In de laatste kolom is aangegeven met ingang van welke release deze functie beschikbaar is. Indien hier niets staat is het release 3.0.

Byte1	Byte2	Byte3	Byte4	Actie	Release
0					
1	block	level		KickStart blok <block> op niveau <level>	
2	block	mode		Stel <block> in op <mode>	
3					
4	block	options		Unlink <block> met <options>	
5	block	source	options	Kopieer/link <source> naar <block> met <options>	
6	block	speed	ticks	Versnel blok <block> naar <speed> met interval <ticks> (analoog)	
7	block	speed	ticks	Versnel blok <block> naar <speed> of 0 (afhankelijk wat het eerst optreedt) met interval <ticks> (analoog)	
8	comm	out32	param	Geef OM32 commando <comm><output><param> voor uitg 0..255	
9	comm	out32	param	Geef OM32 commando <comm><output><param> voor uitg 256..511	
10	device	ticks		Zet RM-H magneetartikel <device> in stand 0, pulsduur = <ticks>	
11	device	ticks		Zet RM-H magneetartikel <device> in stand 1, pulsduur = <ticks>	
12	dev32	ticks		Zet PM32 magneetartikel <device> in stand 0, pulsduur = <ticks>	
13	dev32	ticks		Zet PM32 magneetartikel <device> in stand 1, pulsduur = <ticks>	
14	sw-l	sw-h		Ontkoppel schakelaar (sw-h)(sw-l) van Event Action	
15	sw-l	sw-h	EA-nr	Koppel schakelaar (sw-h)(sw-l) aan <EA>	
16	EA-nr	flags		Zet <flags> van <EA>OFF entry, laat andere ongewijzigd	
17	EA-nr	flags		Zet <flags> van < EA >ON entry, laat andere ongewijzigd	
18	EA-nr	flags		Reset <flags> van < EA >OFF entry, laat andere ongewijzigd	
19	EA-nr	flags		Reset <flags> van < EA >ON entry, laat andere ongewijzigd	
20	EA-nr	flags		Inverteer <flags> van < EA >OFF entry, laat andere ongewijzigd	
21	EA-nr	flags		Inverteer <flags> van < EA >ON entry, laat andere ongewijzigd	
22	EA-nr	flags		Laad entry van < EA >OFF met <flags>	
23	EA-nr	flags		Laad entry van < EA >ON met <flags>	
24	output			Zet uitgang <output>	
25	output			Reset uitgang <output>	
26	output			Inverteer uitgang <output>	
27	output	output	output	Multibit clear/set, bit 7 van <output> geeft gewenste stand ***!Zie noot!***	
28	output	ticks		Puls uitgang <output> gedurende <ticks>	
29	output	ticks		Knipper uitgang <output> met periode <ticks> (<ticks = halve periode)	
30	output	ticks		Flash uitgang <output> met interval <ticks>	
31	output	ticks		Randomize uitgang <output> met interval <ticks>	

Toelichting:

block, source:	0..127
speed:	0..63, bit 6 = polariteit
level:	0..63
options (unlink):	bit 0: maak 0, bit 1: unlink up
options (Kop/link):	bit 0: maak permanent (link), bit 1: inverter polariteit
ticks:	0..255
output:	0..127
out32:	0..255
device:	1..127
dev32:	0..255
EA-nr:	0..255
sw-h.sw-l:	0..2047 (sw-l = 0..255, sw-h = 0..7)
Mutibit clear/set:	Gewenste stand wordt gegeven door bit 7, dus ≥ 128 is aan, ≤127 is uit Indien slechts 2 bytes gebruikt: maak 3^e output gelijk aan 2^e, 3^e wordt dan niet uitgevoerd.
ticks:	Tijd/Interval in eenheden van 1/60 sec

4 Initialisatie en configuratie

Om initialisatiecommando's en configuratieregels naar Dinamo te sturen zijn een aantal programma's beschikbaar. Deze lezen een configuratiefile (tekstbestand) in die voldoet aan de volgende afspraken:

Regels die beginnen met een # gevolgd door 1 t/m 7 getallen sturen een normaal commando naar Dinamo. Het eerste getal is het commando, de volgende getallen zijn de parameters behorende bij dit commando. De commando's zijn de commando's zoals genoemd in de tabel in paragraaf 2.
De getallen mogen gevolgd worden door een willekeurige tekenreeks, mits niet beginnend met een getal. Deze tekenreeks wordt gezien als commentaar en genegeerd.

Voorbeeld:

 # 1 1 12 Stel HFI level in op 12

Regels die beginnen met een : (dubbele punt) sturen een Event Action configuratie. Deze regels moeten er als volgt uit zien:

: <state><EA-nr><flags> <byte1><byte2><byte3><byte4> <byte1><byte2><byte3><byte4>

: = teken dat het een EA (Event Action) configuratieregel is
<state> = 0 (OFF) of 1 (ON)
<EA-nr> = 0..255
<flags> = 1 (voer EA1 uit bij switch event), 2 (voer EA2 uit bij switch event),
4 (voer EA1 uit bij virtuele uitgang), 8 (voer EA2 uit bij virtuele uitgang),
16 (silent mode)
'Silent mode' = Er wordt geen bericht naar de PC gestuurd van het switch event

De 2 groepen <byte1><byte2><byte3><byte4> vormen de 2 acties die bij dit EA horen

Flags mogen worden gecombineerd door optellen, bijvoorbeeld 4 + 8 = 12 betekent "voer EA1 en 2 uit bij Virtuele Uitgang"

Ook de configuratieregels mogen gevolgd worden door commentaar, mits niet beginnend met een getal.