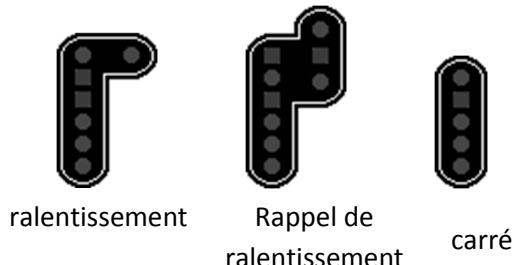


Notice avancée du programme Signaux_complexes

Cette notice décrit le fonctionnement du programme client signaux complexes GL.

Exemples de signaux complexes :



Rappel de
ralentissement combiné à
un avertissement

Matériel nécessaire :

- le kit client exemple signaux_complexes qui pourra être utilisé comme point de départ, téléchargeable ici
- CDM rail V5.1 mini.
- décodeur de feux « led dekoder » de **DigitalBahn** équipé de son logiciel « led_signal_10 » ou un décodeur **CDF** ou **LEB** ou **LDT-DEC-SNCF** ou **Leb-Modélisme** ou **NMRA** ou **UniSemaf** ou **SR**
- L'environnement de développement Delphi7
- éventuellement, un programmateur de PIC pour reprogrammer les adresses du décodeur de digital bahn.

<i>Introduction</i>	4
<i>Exemple d'implantation</i>	5
<i>Exemples de situations</i>	6
<i>Règles de présentation des signaux combinés</i>	7
<i>Règles de présentation des feux directionnels</i>	8
Exemple d'implantation	8
<i>Principe de pilotage</i>	9
Détail des conditions	13
Utilisation des détecteurs pour les mémoires de présence de train	15
Utilisation des détecteurs	15
Différences entre actionneurs et détecteurs	15
Actionneur (détection logicielle)	15
Détecteur (détection électrique)	15
<i>Feux directionnels / TIV</i>	17
Décodeurs Digital Bahn	17
Décodeurs CDF	18
<i>Restrictions et spécificités du programme client pour les signaux</i>	19
<i>Configuration du décodeur Digitalbahn pour les signaux complexes</i>	20
Configuration par programmateur	20
Configuration « manuelle » du décodeur de signal digitalBahn (si on ne dispose pas de programmateur)	22
Fichiers PIC pour le décodeur DigitalBahn	23
<i>Codes commandes pour le décodeur digitalBahn</i>	23
<i>Logiciels PIC pouvant équiper le décodeur digital Bahn</i>	23
<i>Mise en place du décodeur DigitalBahn</i>	24
<i>Utilisation du décodeur CDF80108007S pour les signaux complexes</i>	25
<i>Utilisation du décodeur LDT-DEC-SNCF pour les signaux complexes</i>	27
<i>Utilisation du décodeur LEB pour les signaux complexes</i>	28
<i>Configuration du décodeur Digitalbahn pour les feux directionnels ou les TIV</i>	30
<i>Description logicielle du programme client SIGNAUX_COMPLEXES</i>	30
Modification du programme avec Delphi 7	31
Installation des composants sockets (TClientSocket et TServerSocket) pour Delphi7	31
Installation de Rad Studio 12 (Delphi12)	Erreur ! Signet non défini.
Installation des composants sockets (TClientSocket et TServerSocket) pour RAD Studio	32

Installation du composant MScomm32 pour Rad Studio	33
Debuggeur	35
Installation du composant MScomm32 pour Delphi7	36
Défaillances possibles sous DELPHI7	36
Compilation en 32 ou 64 bits avec RadStudio 12 :	39
Généralités sur la détermination des routes sur les évènements détecteurs	40
Connexion aux interfaces en XpressNet en port COM ou USB	41
Gestion de la liaison XpressNet ou DCC++ (mode autonome)	41
Gestion du calcul de progression des trains (routes)	42
Procédures et fonctions	46
Chronologies d'appels du pilotage des trains en mode autonome	47
Fonctionnement de l'échange de données entre le client et le serveur CDM Rail	49
Structure du fichier ConfigGenerale.cfg	50
Section principale	51
Codification des aiguillages simples	52
Aiguillage Triple	54
Exemple de modélisation de deux d'aiguillages triples	55
Modélisation des TJD	55
Modélisation des TJD et des TJS	55
Codification d'une TJS	58
Section de modélisation des feux	59
Ligne de modélisation :	59
Exemple d'un feu avec conditions supplémentaires sur le carré	62
Section Actionneurs	63
Actionner une fonction F d'une locomotive (F1 à F12)	63
Actionner un passage à niveau à une ou plusieurs voies	64
Actionner un accessoire depuis un actionneur	64
Section décodeurs personnalisés	64
Section DCC++	65
Section Trains	65
Section Trains	65
Section Périphériques	65
Annexe 1 : interface permettant l'utilisation de signaux avec commun au -	67
Annexe 2 : câblage des décodeurs pour un feu complexe et feux directionnels (décodeurs digitalbahn)	68

Annexe 3 : guide de choix des signaux et des décodeurs	69
Annexe 4 - Liste des états d'un signal SNCF complet à 10 feux	72

Introduction

Cette notice est un complément technique de niveau 2 qui fait suite à la notice d'utilisation des signaux complexes. Elle aborde et décrit le programme client des signaux complexes GL.

En mode connecté avec CDM Rail, le programme client signaux complexes calcule les états des signaux complexes en fonction des informations de détection du réseau transmis depuis CDM rail. En mode autonome, il le fait d'après les informations des détecteurs transmis par la centrale via l'interface.

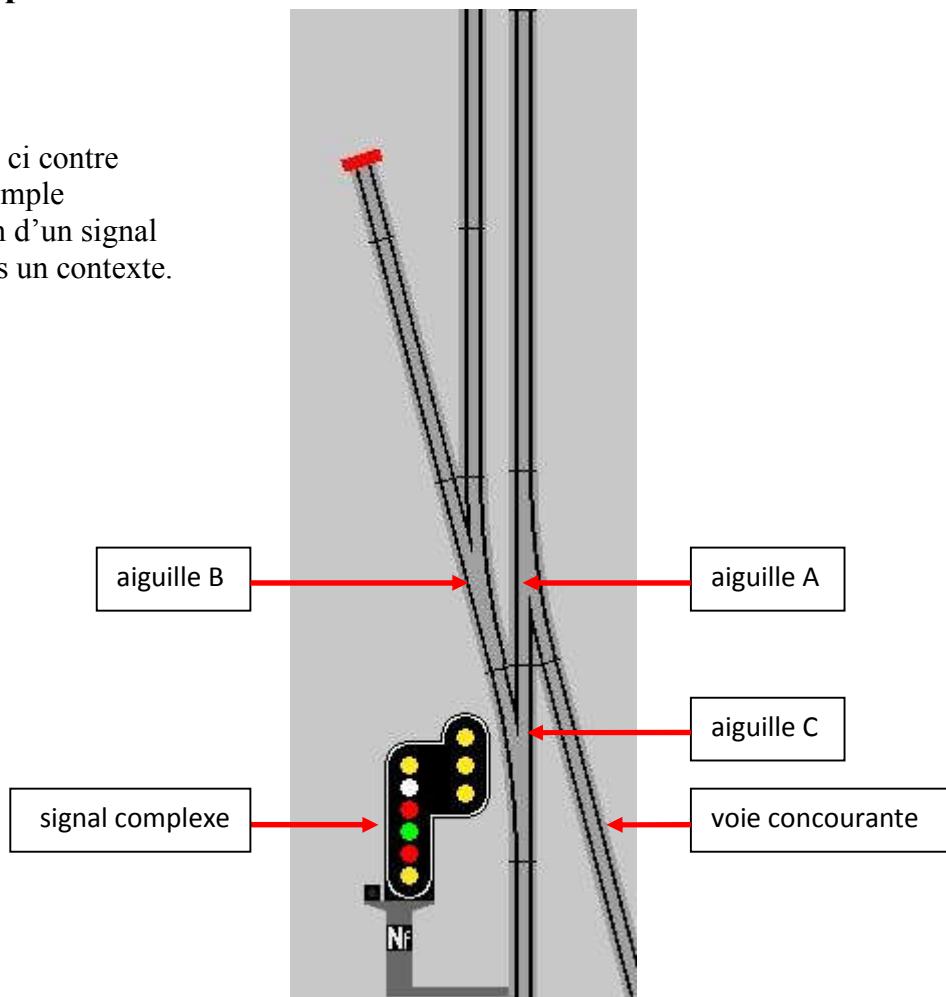
Il met en mémoire les détecteurs actionnés, puis tente de les associer pour vérifier que deux détecteurs actionnés l'un après l'autre l'ont été par le même train ou non.

Les mémoires de zone sont des associations de deux détecteurs contigus qui informent de la présence train entre deux détecteurs. Ces mémoires sont calculées automatiquement par le programme client (variable tableau `MemZone[detect1,detect2]`). Cette mémoire signifie qu'un train circule entre le front descendant du détecteur1 vers le front descendant du détecteur2.

Ces mémoires de zone servent à afficher le sémaphore sur un signal lorsque la mémoire de zone aval d'un signal passe à 1. Un aiguillage dévié en aval d'un signal et avant un signal suivant affichera un rappel de ralentissement (si toutefois le feu est de forme n°9). Tous les autres états d'un signal (jaune, jaune clignotant, vert, ralentissement) sont générés en fonction de l'état du signal suivant. S'il n'y a pas de signal suivant complexe sur un signal considéré, ce signal n'affichera que rouge ou vert.

Exemple d'implantation

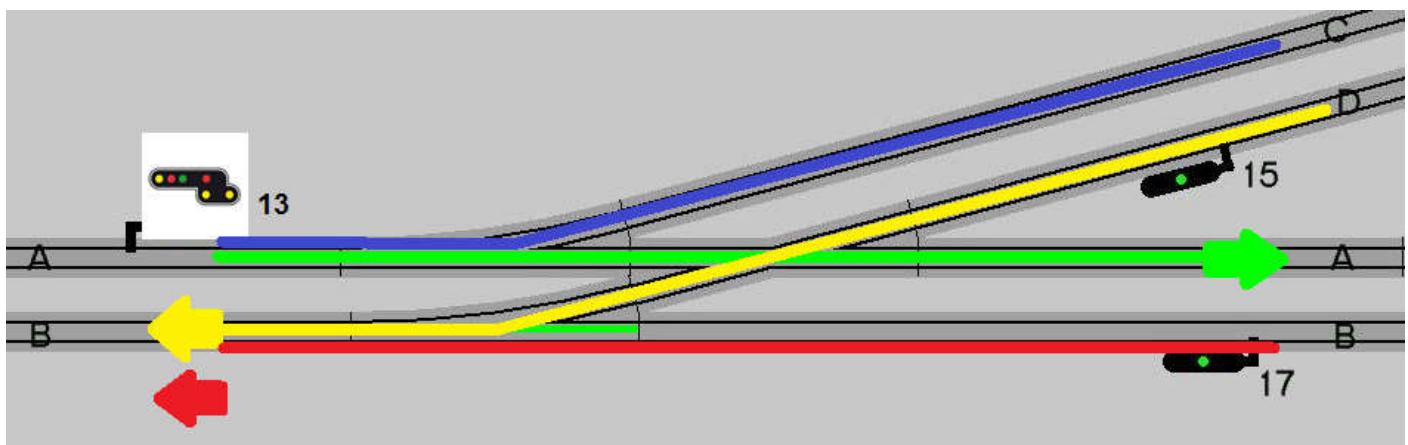
Le synoptique ci contre montre un exemple d'implantation d'un signal complexe dans un contexte.



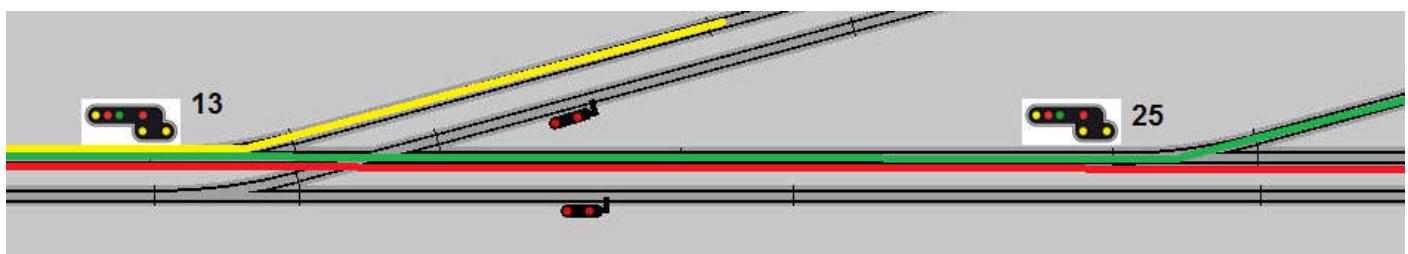
présentation du signal	règles de présentation du signal
carré	-aiguille A pas en direct -signal de la voie concourante ne présentant pas de carré (voie ouverte) -pas de présence train sur les cantons précédant le signal
sémaphore	présence train sur les cantons précédant le signal et train présent sur le canton suivant le signal (tenir compte de la position des aiguilles B et C) (fonctionnement en BAL)
avertissement	-signal suivant au carré ou au sémaphore si présence train sur les cantons précédant le signal (fonctionnement en BAL) -aiguille C déviée, aiguille B droite (vers voie en buttoir)
avertissement clignotant	signal suivant à l'avertissement ou au ralentissement si présence train sur les cantons précédant le signal – (fonctionnement en BAL)
ralentissement 30 ou 60	aiguille distante prise en pointe déviée
rappel 30 ou 60	aiguilles locales prises en pointe déviées : C ou (C et B) donc C
rappel 30 ou 60 + avertissement	-aiguille C déviée, aiguille B droite : le train est dirigé vers la voie de garage -aiguille C déviée, B déviée et signal suivant au carré ou au sémaphore -aiguille C déviée, B déviée et aiguille distante déviée
rappel 30 ou 60 + avertissement clignotant	aiguille C déviée, B déviée et signal suivant à l'avertissement
manœuvre (blanc ou clignotant)	le feu blanc est provoqué par le poste d'aiguillage, il s'agit d'une condition manuelle. Pour Signaux complexes, il est provoqué par la coche «demande FB» du signal
voie libre	Aiguilles C et A en direct et voie libre sur les 2 cantons en avant, sans déviation d'aiguillage distant.

Exemples de situations

1. Un train circule sur la voie A dans le sens de la flèche verte. Le signal 15 doit être au carré et non au sémaphore.
2. Un train circule de la voie D vers la voie B dans le sens de la flèche jaune. Les signaux 13 et 17 doivent être au carré.
3. Un train circule sur la voie B dans le sens de la flèche rouge. Le signal 15 doit être au carré et non au sémaphore
4. Un train circule de la voie A à la voie C (trajet bleu). Le signal 13 doit être sur rappel de ralentissement 30 ou 60 km/h si l'aiguillage ne peut être franchi à plus de 30 ou 60 km/h. Cela implique que le signal précédent doit être sur ralentissement.



5. Un train circule sur le parcours jaune. Le signal 13 doit présenter un rappel de ralentissement et éventuellement un avertissement ou un jaune clignotant en fonction du feu suivant sur le trajet jaune. Le feu précédent doit présenter un ralentissement.
6. Un train circule sur le parcours vert. Le signal 13 doit présenter un ralentissement, le signal 25 un rappel de ralentissement et éventuellement un avertissement ou un jaune clignotant en fonction du feu suivant sur le trajet vert.
7. Un train circule sur le parcours rouge. Ce sont les signaux de BAL qui s'appliquent. Les ralentissement et les rappel de ralentissement sont éteints.



Autres exemples de situations sur ces sites :

http://www.ligneduhautbugey.fr/bal_gare.html
<http://snCF.ratP.free.fr/circulationssnCF.htm>

Règles de présentation des signaux combinés

Certaines combinaisons de signaux combinés ne sont pas présentées. Par exemple, si un train doit franchir plusieurs aiguillages distants déviés, le mécanicien devrait rencontrer un signal avec un ralentissement et un rappel de ralentissement présentés simultanément. Cette combinaison est remplacée par un rappel de ralentissement avec un avertissement :



L'allumage d'un ralentissement 60 ou d'un rappel de ralentissement n'éteint pas l'avertissement ni l'avertissement clignotant.

Inversement : un ralentissement 30 efface l'avertissement.

Le jaune clignotant n'efface pas le ralentissement 60.

Inversement, le jaune fixe efface le ralentissement 30 ou le ralentissement 60

Ainsi, seules les combinaisons suivantes sont autorisées :



Rappel 30 + avertissement fixe OU
Rappel 30 + avertissement clignotant OU
Rappel 60 + avertissement fixe OU
Rappel 60 + avertissement clignotant

Ralentissement 60 + avertissement clignotant
(tous feux clignotants)

Les décodeurs gèrent automatiquement ces combinaisons. Les restrictions sont les suivantes :

Décodeurs LDT : la présentation des rappels 30 ou 60 combinée avec l'avertissement ou l'avertissement clignotant n'est pas possible.

Décodeurs CDF : représentation maximale : 8 feux à 6 états. Pour pouvoir utiliser les combinaisons, il convient de considérer qu'il y a 2 signaux séparés sur le même panneau : un signal principal avec carré, sémaphore, avertissement et voie libre et un signal auxiliaire Ral ou Rap Ral .

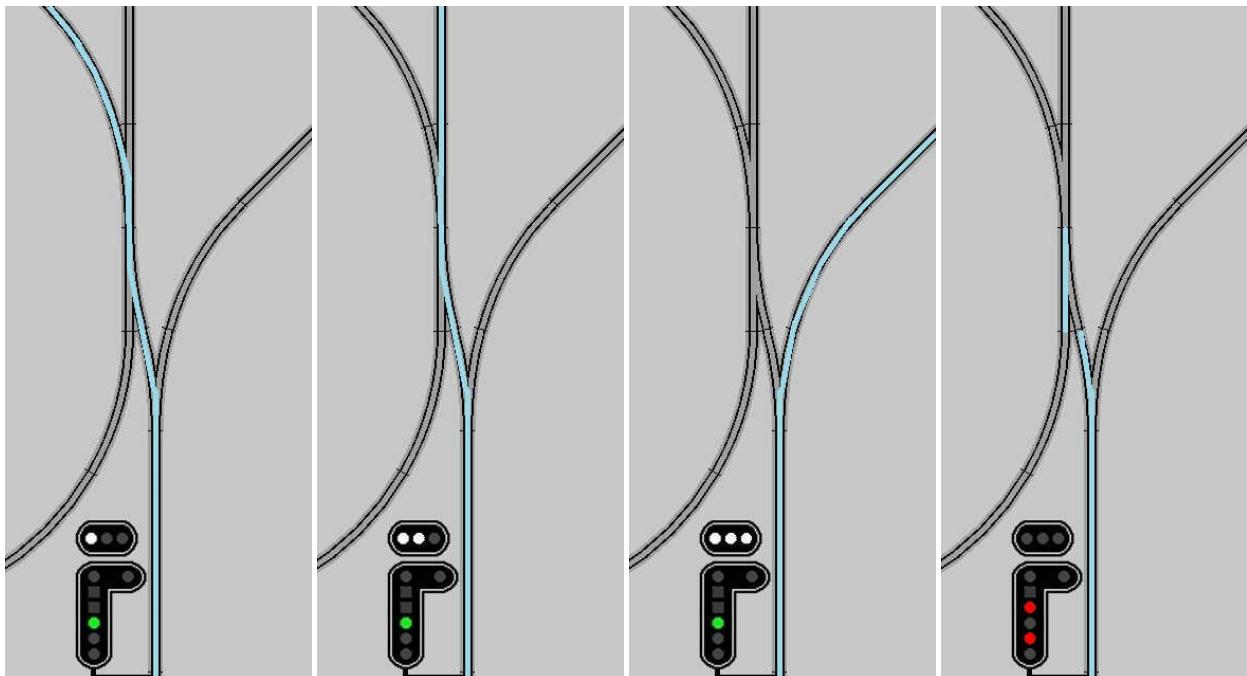
Décodeurs Digital Bahn : tous les états sont possibles, mais il faut gérer l'extinction de l'avertissement sur un ralentissement 30 (géré dans l'organigramme du programme client)

Décodeurs LEB : tous les états sont possibles. Ce décodeur nécessite une remise à 0 de ses entrées après les avoir positionnées.

Règles de présentation des feux directionnels

Exemple d'implantation

Les signaux directionnels indiquent la direction géographique de l'itinéraire. La référence de direction est à gauche. Les feux directionnels ne s'allument que si la position des aiguillages est cohérente.



1^{ère} direction à partir de la gauche : un feu directionnel allumé

2^{ème} direction à partir de la gauche : deux feux directionnels allumés

3^{ème} direction à partir de la gauche : trois feux directionnels allumés

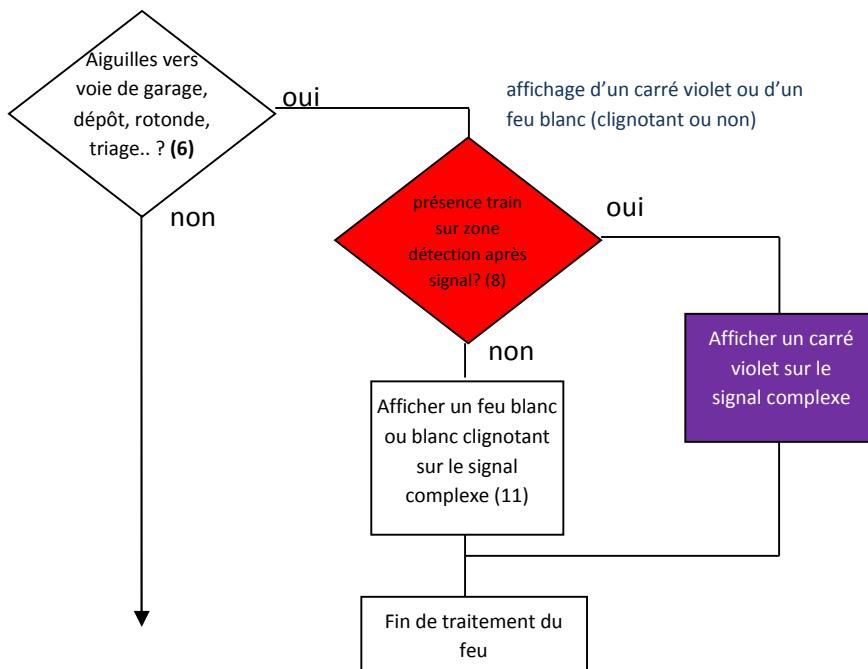
Direction invalide car la position de l'aiguillage pris en talon n'est pas positionnée dans le sens du déplacement : le signal présente donc un carré et l'indicateur de direction est éteint.

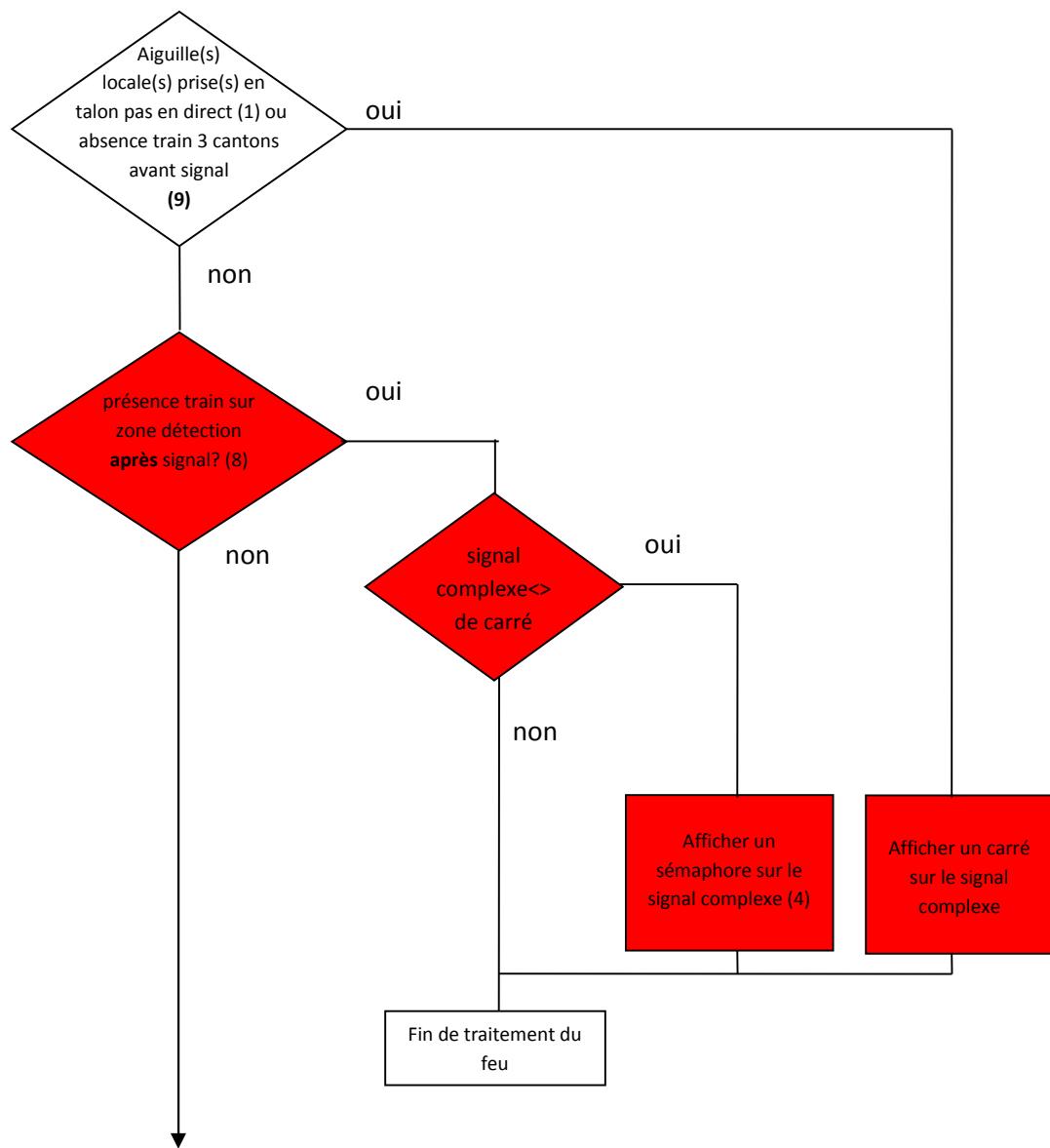
Le pilotage des sorties se fait en fonction des aiguillages qui dirige le train vers la position géographique désignée par le signal directionnel. Le nombre de feux directionnels dépend du nombre de positions géographiques de destination.

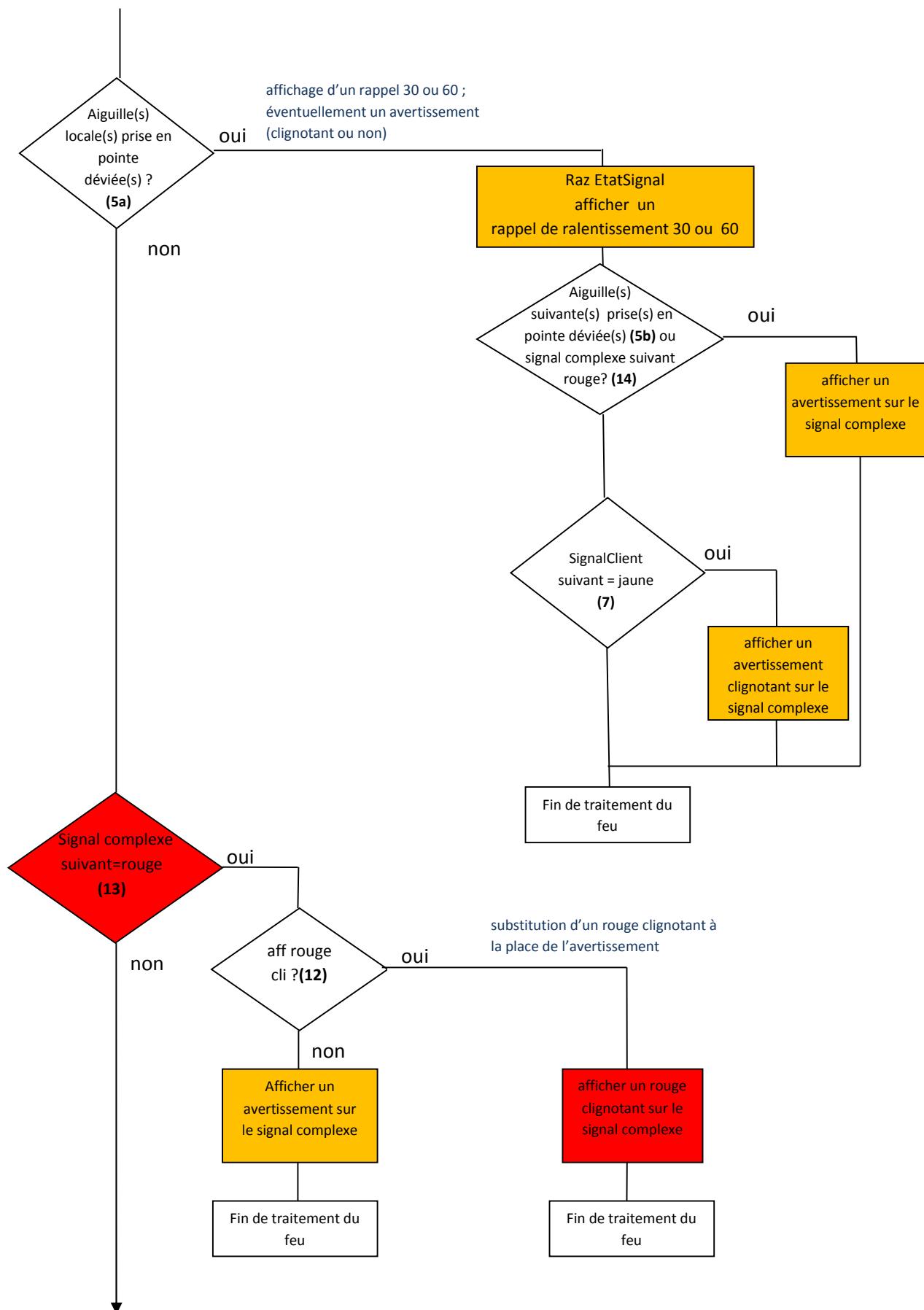
NB. Le feu complexe présenté ci-dessus pour exemple n'a pas la forme permettant de représenter un rappel de ralentissement car les aiguillages pris en pointe permettent dans ce cas de les franchir à plus de 60 km/h. Un TIV conditionnel doit donc être implanté avant les aiguillages.

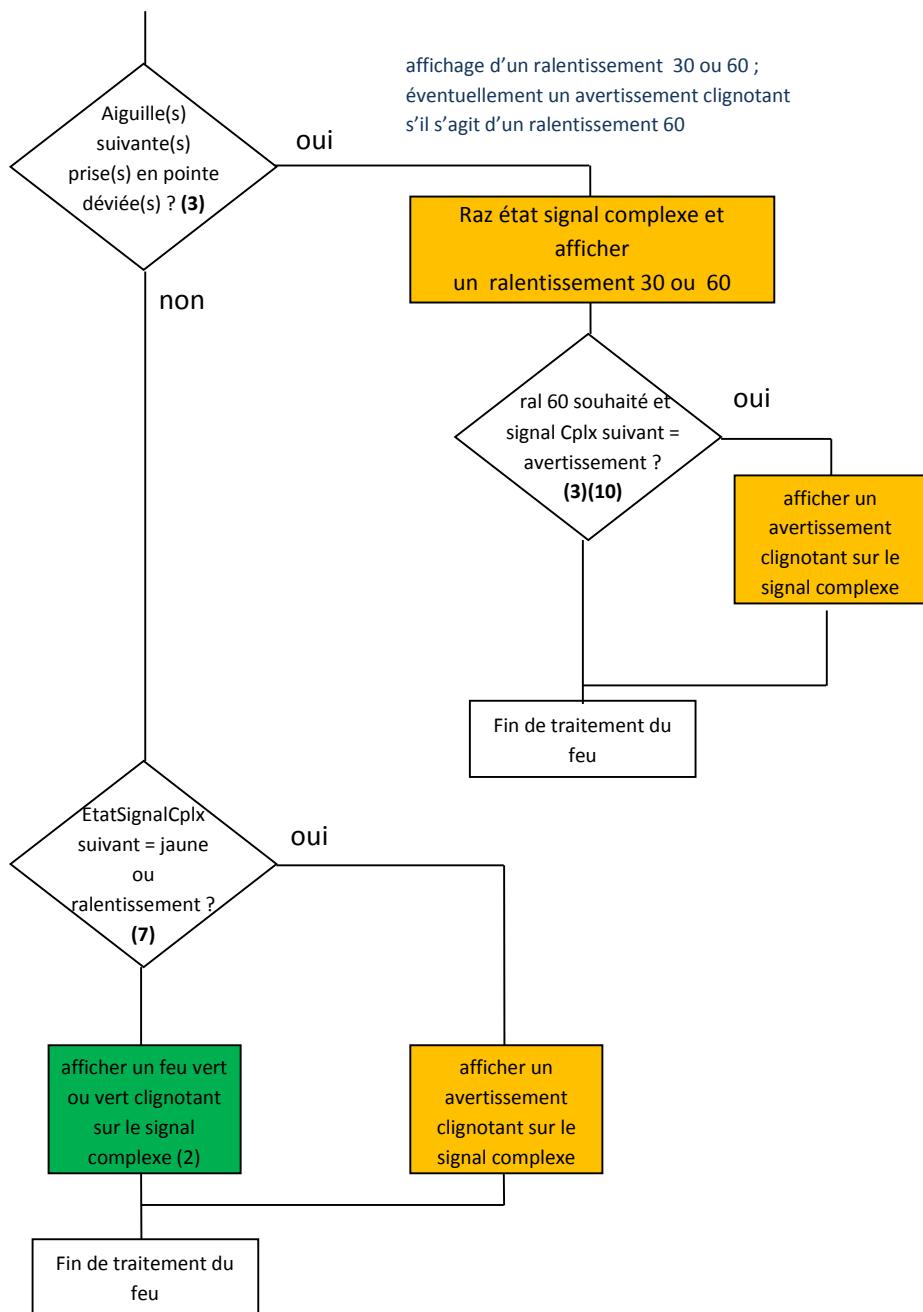
Principe de pilotage

Le principe de pilotage d'un feu complexe complet (c'est à dire doté d'un ralentissement et/ou d'un rappel de ralentissement et/ou d'un carré et/ou d'un carré violet) est le suivant. Le travail du programme client consiste à regarder l'état des détecteurs pour piloter le signal complexe associé via le décodeur, en fonction de la position du ou des aiguillages distants ou à proximité et des feux suivants, et également de la présence ou non d'un train sur les cantons précédant le signal à gérer. Ces organigrammes sont utilisés dans la procédure Maj_Feu (int AdrFeu) . L'organigramme de pilotage pour UN signal complexe est décrit ci-dessous :









Détail des conditions

- (1)** Liste d'aiguillages prises en talon en position bloquantes pour le signal (qu'elles soient déviées ou droites.)
Laisser vide ou mettre une adresse de signal inexistante s'il n'y a pas d'aiguille après le signal ou si le signal ne possède pas de carré.
- (2)** le feu vert clignotant peut être affiché si on coche « Feu vert clignotant » dans la configuration du signal.
- (3)** traitement seulement si le feu dispose d'un ralentissement.
- (4)** le sémaphore clignotant peut être affiché en gare (évitant ainsi l'arrêt en gare) ou avant une rampe. Il nécessite de faire circuler le train à 15 km/h jusqu'à la queue du train suivant (ce qui est actuellement impossible car le feu CDM correspondant est rouge ce qui arrête obligatoirement le train, il faut donc le substituer à un sémaphore). Le sémaphore clignotant peut également annoncer un carré sur un canton court. (voir note 13). le feu rouge clignotant peut être affiché si on coche « demande feu rouge clignotant » dans la configuration du signal.
- (5a)** traitement seulement si le feu dispose d'un rappel de ralentissement : liste d'aiguillages locaux pris en pointe en position « déviée ».
- (5b)** si l'aiguille suivante est déviée ou le signal suivant est rouge (sémaphore ou sémaphore clignotant ou carré ou carré violet) ou si le train sera dirigé vers une voie en buttoir.
- (6)** Liste d'aiguillages avec leurs positions amenant le train en face du signal vers une voie de garage.
Cas d'un signal sur une voie amenant le train sur un buttoir :
- si présence d'un train sur la voie en buttoir, il s'agit d'une manœuvre, donc afficher un feu blanc ou un feu blanc clignotant. Pour détecter la présence d'un train sur une voie de garage, utiliser un détecteur et non un actionneur (évidemment il faut utiliser des essieux graphités ou une rame avec feux)
- si pas de train sur la voie en buttoir, le feu doit présenter un avertissement.
- ce peut être également une condition manuelle du programme client qui bascule le feu au blanc ou blanc clignotant.
- (7)** Si on ne désire pas gérer l'avertissement clignotant, ne pas traiter le cas. L'avertissement clignotant précède soit un avertissement soit un ralentissement 60 si le canton est court.
- (8)** présence train dans la zone de détection après signal : utiliser les mémoires de zone de détection. Voir page suivante.
- (9)** Présence d'un train sur les 3 cantons précédents le signal pour afficher un carré sur un signal pouvant afficher le carré. Il faut utiliser une mémoire de présence en entrée et en sortie de la zone pour détecter la présence d'un train. L'actionneur d'entrée ou le détecteur monte une mémoire de présence dans le programme client et l'actionneur ou le détecteur de sortie fait retomber cette mémoire. *Voir page 15 « Utilisation des détecteurs pour les mémoires de présence de train » et utilisation des détecteurs.*
- (10)** Le test doit être effectué si l'on désire afficher un rappel 60 ou un ralentissement 60 et non 30 sur le signal complexe (car la règle de présentation de l'avertissement est différente en 30 ou en 60).
- (11)** Le feu blanc clignotant est à utiliser pour une manœuvre sur une section courte ou un parcours limité en distance n'autorisant pas le départ en ligne. Il est plus généralement déclenché par une condition manuelle qui verrouille d'autres signaux au rouge pour permettre au train de faire sa manœuvre en sécurité. Par contre, le feu blanc peut être suivi d'un feu vert.
- (12)** Condition d'affichage du sémaphore clignotant (rouge cli) : si on désire que le signal complexe affiche un rouge cli au lieu d'un avertissement. La seule possibilité actuellement d'afficher un sémaphore clignotant est de cocher "Feu rouge clignotant" dans les propriétés du signal.
- (13)** Si le signal suivant est rouge ou violet (sémaphore ou sémaphore clignotant ou carré ou carré violet) ou si le train sera dirigé vers une voie en buttoir.

Note 1 : signal suivant signifie le signal suivant le signal considéré d'après le cheminement du ou des aiguilles.

Même remarque pour « aiguille suivante »

Note 2 : pour ne faire figurer qu'un ralentissement ou un rappel sur un décodeur signal bahn sur lequel figure déjà un avertissement, il faut demander l'affichage d'un carré, vert, ou sémaphore, puis demander l'affichage du ralentissement ou du rappel et réciproquement. Les deux demandes doivent être espacées de 400ms au moins, sinon la première demande ne sera pas prise en compte.

Note 3 : on peut utiliser des actionneurs pour activer et désactiver des mémoires de présence train. Les détecteurs ne sont pas utilisables en mode simulation. D'autre part, les détecteurs ne

sont pas forcément implantés dans des endroits stratégiques, alors que les actionneurs peuvent être implantés à souhait.

Note 4 : L'adresse de configuration des actionneurs doit être indépendante de tout élément de pilotage du réseau DCC (aiguillages, signaux complexes, décodeurs etc)

Utilisation des détecteurs pour les mémoires de présence de train

Les mémoires de zones de présence de trains servent d'une part dans le programme client pour suivre un train et afficher un carré en cas d'absence de train avant le signal sur 3 cantons. Elles servent d'autre part à séquencer les signaux.

Utilisation des détecteurs

L'utilisation des détecteurs est possible en mode TCO (mode RUN sans trains). C'est pourquoi il faut les privilégier au détriment des actionneurs. Les détecteurs sont l'image de la présence du train par un détecteur physique sur le réseau.

Exemple : MemZone[527,520]=TRUE signifie présence train de la fin de la zone de détection 527 vers la fin de la zone de détection 520.

Les actionneurs ne sont utilisés dans le programme client que pour déclencher des actions sur les fonctions F des locomotives ou gérer des passages à niveaux.

Différences entre actionneurs et détecteurs

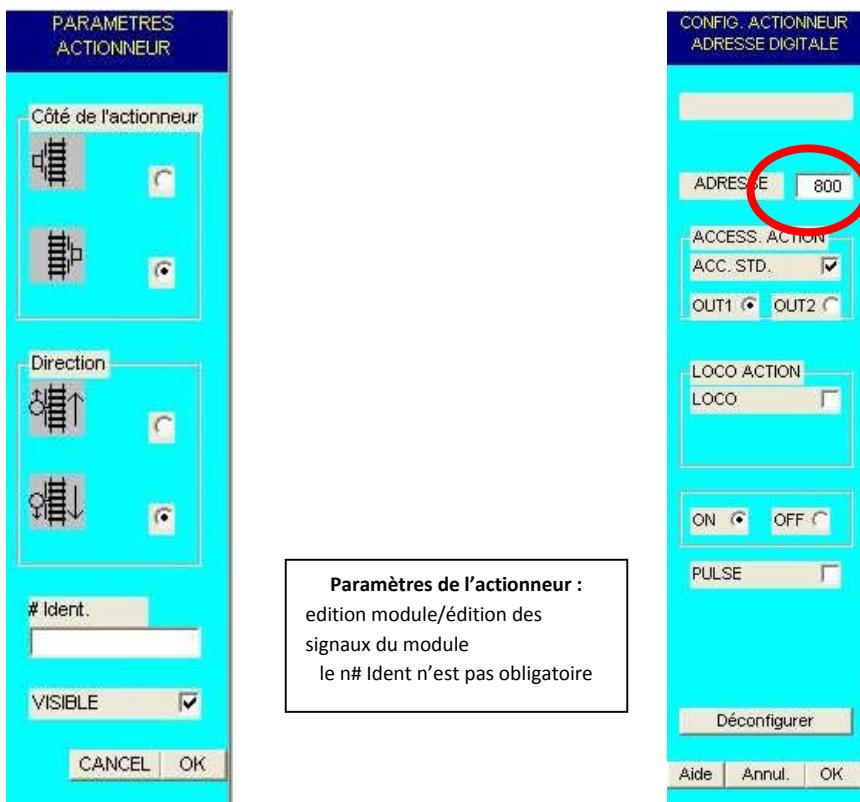
Actionneur (détection logicielle)

- Les actionneurs ne sont utilisables qu'en mode RUN avec au moins un train.
- Activation dynamique (au passage d'un train en mouvement).
- Sensibilité au sens de passage du train.
- Permet la détection en un point.
- Utilisable en mode simulation ou avec centrale allumée.

Détecteur (détection électrique)

- les détecteurs sont utilisables en mode RUN avec ou sans train (mode TCO)
- Activation statique et dynamique (permet la détection d'un train même arrêté)
- Insensible au sens de passage train (ne permet donc pas la détection du sens du train)
- Si le dernier essieu du train est graphité ou avec prise de courant pour feu de fin de convoi, il y aura une double détection du même train.
- Permet la détection sur toute la zone de détection.
- Utilisable uniquement centrale allumée.

Ci dessous, les tableaux de configuration des **actionneurs** dans CDM rail. Le programme client identifie l'actionneur par son **adresse** et non pas par son numéro #ident, qui n'est pas obligatoire.



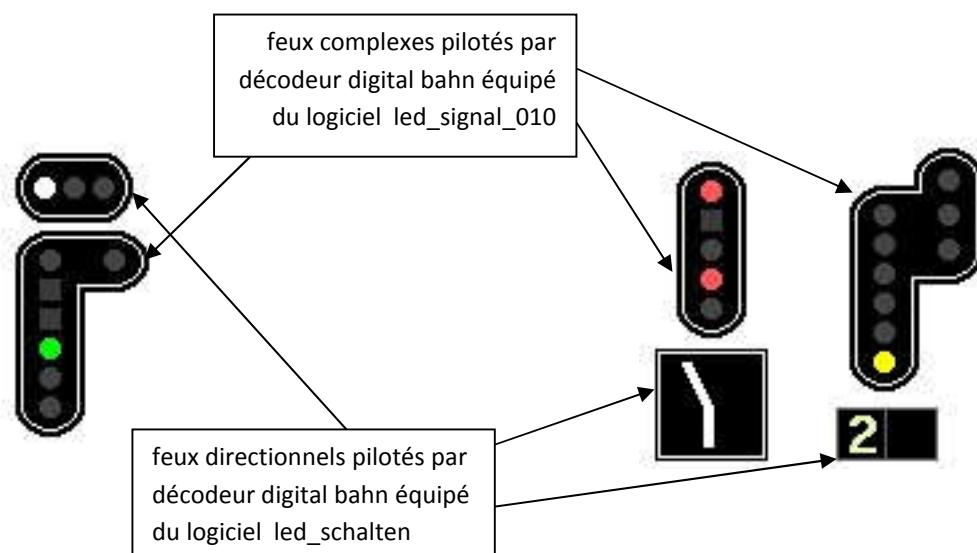
configuration de l'actionneur :
configuration/configuration
d'actionneur.
renseigner l'adresse qui sera
utilisée dans le programme client.
De base, un actionneur sert à
piloter un accessoire comme ici, la
coche de ACC STD est présente et
envoie une commande sur la sortie
OUT 1 de l'adresse 800.

Feux directionnels / TIV

Décodeurs Digital Bahn

Pour utiliser un tableau indicateur de feux directionnels ou un TIV conditionnel ou tout autre type de pancarte lumineuse à plusieurs états, il faut l'équiper de son propre décodeur, équipé du logiciel pic « **led schalten** ». Ce programme transforme le décodeur de leds en décodeur de 10 sorties allumables individuellement en mode « marche/arrêt ». Le décodeur ainsi équipé de ce logiciel occupe 10 adresses. La configuration des adresses du feu directionnel du fichier HEX (**led_16f684_schalten_355_dcc.hex** pour la version DCC) se fait à l'aide de **HEX_MANIPU.EXE**

Feux directionnels :

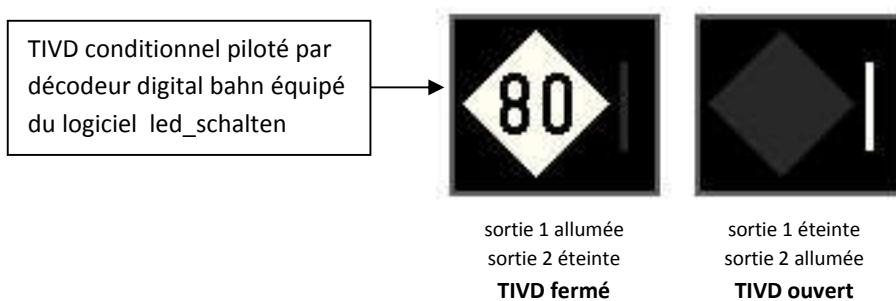


Exemple :

`Envoi_DirectionBahn(372,0)` : éteint le signal directionnel d'adresse 372

`Envoi_DirectionBahn(372,2)` : allume 2 voyants sur le signal directionnel d'adresse 372

TIVD :



NB : TIVD=tableau indicateur de vitesse à distance, en annonce d'un aiguillage distant pris en pointe dont la vitesse de franchissement est différente de 30 ou 60 km/h pour lequel on aurait implanté un signal complexe.

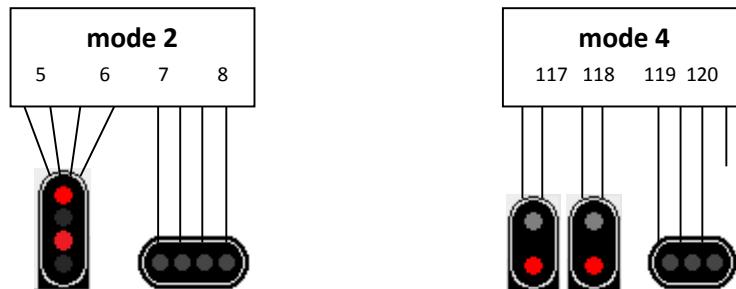
Le TIVD se pilote avec la procédure `Envoi_DirectionBahn`.

le fichier pic pour le protocole DCC peut se télécharger ici :

http://www.digital-bahn.de/bau_led/led_schalten.htm

Décodeurs CDF : Procédure pour utiliser un tableau indicateur de feux directionnels ou un TIV conditionnel à 2 états ou tout autre type de pancarte lumineuse à plusieurs états simultanés :

exemples :



Dans le programme client, la procédure de pilotage des décodeurs CDF s'appelle ***envoi_directionCDF***. Cette procédure fournit une méthode de pilotage suivant le câblage standardisé ci dessus inspirée de la documentation CDF. Pour d'autres câblages, il faut adapter le code de pilotage dans la procédure. Le CV 37 du décodeur indique d'une part qu'il s'agit d'un signal directionnel et permet de renseigner le nombre de feux utilisé sur le signal directionnel donc le nombre de directions possibles.

appel de la procédure :

Envoi_DirectionCDF(Adresse,Code)

Adresse : adresse DCC de base du signal directionnel sur le bus

Code : nombres de voyants à allumer sur le signal directionnel (0 à 4) à partir de la gauche

Exemple d'utilisation :

<u>Envoi_DirectionCDF(7,0)</u>	; Eteint le signal directionnel d'adresse 7
<u>Envoi_DirectionCDF(7,2)</u>	; Allume 2 voyants sur le signal directionnel d'adresse 7
<u>Envoi_DirectionCDF(7,3)</u>	; Allume 3 voyants sur le signal directionnel d'adresse 7
<u>Envoi_DirectionCDF(7,4)</u>	; Allume 4 voyants sur le signal directionnel d'adresse 7
<u>Envoi_DirectionCDF(119,3)</u>	; Allume 3 voyants sur le signal directionnel d'adresse 119

Restrictions et spécificités du programme client pour les signaux

En utilisation avec CDM rail en mode RUN sans itinéraire, les signaux seront positionnés en fonction des aiguillages. C'est à l'opérateur de manœuvrer les aiguillages suffisamment à l'avance avant le passage du train pour que la présentation des signaux soit cohérente.

En utilisation avec CDM rail en mode RUN avec itinéraire(s), les aiguillages sont positionnés 1 canton avant le passage du train, ce qui est trop tard pour l'affichage d'un signal de ralentissement.

Pour utiliser les signaux complexes en mode TCO (mode RUN sans train), les actionneurs ne peuvent pas être utilisés car le train virtuel n'existe pas. Il faut utiliser les détecteurs. Cela concerne la montée des mémoires de présence de train en zone et l'affichage du feu rouge à la retombée de la détection de la zone avant le signal.

Dans CDM, si les aiguillages sont configurés avec l'option « répéter commandes » cochée (configuration/configuration d'aiguillage), les feux pourraient clignoter. Dans ce cas, à l'arrivée d'un convoi, CDM commande à nouveau les aiguillages même s'ils sont bien positionnés si cette option est cochée. La commande des aiguillages réévalue les feux et donc réactualise les signaux.

Pour une rapidité d'exécution optimale, dans CDM, il faut absolument dévalider les deux options de création des logs dans les menu « Comm IP/créer un fichier de log » et « Autoriser le log des évènements de service » (avant de lancer le serveur de l'interface).

Configuration du décodeur Digitalbahn pour les signaux complexes

Le décodeur de DigitalBahn équipé du logiciel pour les signaux complexes peut être configuré manuellement à l'aide du bouton se trouvant sur le décodeur, ou bien par un programmateur de pic, et il doit être programmé avec le fichier **decodeur.X.production.hex** (signal SNCF version DCC avec clignotement 2Hz). Les tableaux ci-dessous décrivent le paramétrage du décodeur pour les signaux complexes en modifiant le fichier HEX du PIC par programmateur ou en modifiant directement manuellement les adresses.

Configuration par programmateur

(plus souple que la configuration manuelle) Le logiciel HEX manipu permet d'attribuer des adresses DCC aux feux que doit piloter le signal.

Dans le logiciel « HEX Manipu » téléchargeable sur le site de Digital Bahn, ouvrir le fichier « HEX » du PIC (bouton « laden ») (choisir l'HEX relatif à votre protocole : DCC ou Motorola) et modifier les adresses de manière à ce qu'elles soient consécutives afin que le pilotage du signal soit compatible avec le programme client. L'exemple montré ici concerne le signal 78. Sauvegarder en cliquant sur «speichern ». Il faudra ensuite utiliser ce fichier HEX ainsi modifié dans le programmateur de PIC pour l'y injecter.

LED-Dekoder - Signal 010 - Signal SNCF		
C - Stop Carré	78	R/G
S - red	79	R/G
(S) - red flash.	80	R/G
VL - green	81	R/G
(VL) - green flash.	82	R/G
Cv - violet	83	R/G
M - white	84	R/G
(M) - white flash.	85	R/G
A - yellow	86	R/G
(A) - yellow flash.	87	R/G
30 R - slow	88	R/G
60 (R) - slow	89	R/G
30 RR - remember	90	R/G
60 (RR) - remember	91	R/G

[www.DIGITAL-BAHN.de](http://www.digital-bahn.de/bau_led/led_signal_010.htm)

Ce tableau présente les 14 états pilotables du signal et affecte chaque fonction à une adresse DCC (ici de 78 à 91).

R/G : permet d'adresser la fonction réalisée à l'adresse indiquée par le bit 1 (vert=2) ou 0 (rouge=1). Laisser vert par défaut pour compatibilité avec Signaux_complexes.

Exemple 1 : Le tableau ci dessus signifie que pour envoyer un carré au signal, il faut envoyer « 2 » à l'adresse DCC 78 car l'adresse est codée en vert. Si elle est codée en rouge (changement par bouton R/G), il faut envoyer « 1 ».

Exemple 2 : Pour envoyer un carré violet, il faut envoyer « 2 » à l'adresse DCC 83.

Ce pilotage est réalisé par le programme client par cette fonction :
`pilote_acc(adresse+5, 2, feu);`

Cette commande envoie « 2 » à l'accessoire du bus DCC **adresse+5 (avec adresse=78)**

L'envoi d'une commande efface la précédente. Voir les exceptions ci dessous en NB.

Pour l'utilisation dans signaux complexes, chaque décodeur doit occuper 14 adresses, c'est-à-dire que toutes les cases de chaque fonction doivent être affectées de façon incrémentale (+1 à chaque adresse), même si elles ne sont pas utilisées.

NB1. Les commandes d'affichage du ralentissement 30 ou 60, d'un rappel 30 ou 60 ou d'un avertissement ne sont pas effacées entre elles.

NB2 : Pour afficher un rappel de ralentissement combiné à un avertissement, il faut envoyer deux commandes.

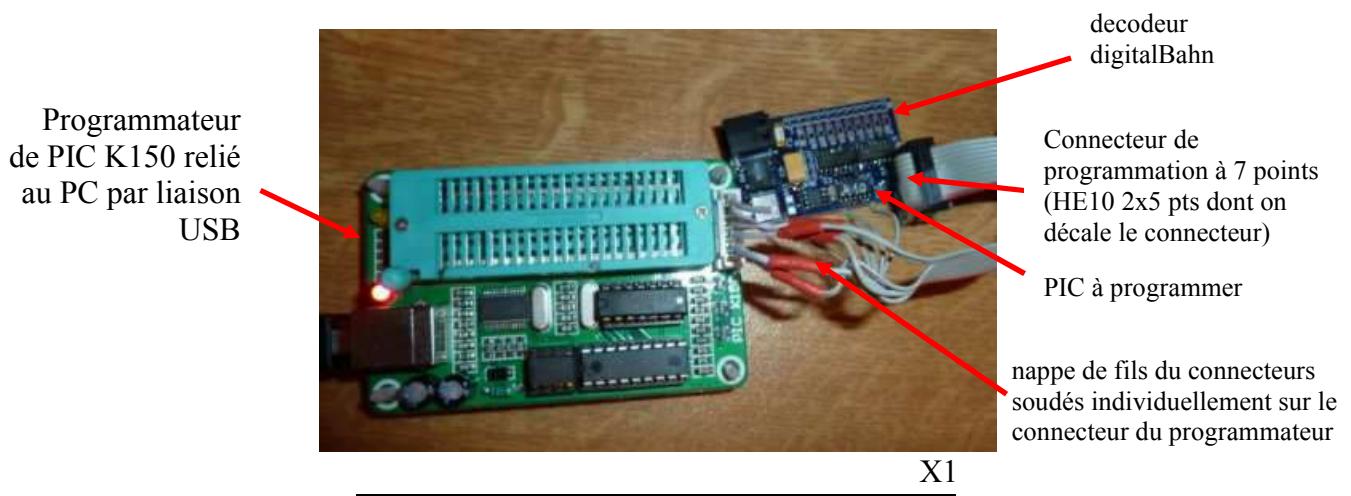
NB3 : Pour afficher un ralentissement 30 sur un signal présentant déjà un avertissement, il faut envoyer un sémaaphore pour effacer le ralentissement 30 ;

idem pour les ralentissemens 30, 60 et avertissement. Ceci est présenté dans les organigrammes.

Configuration du décodeur de signal DigitalBahn par programmateur (suite)

Pour envoyer la configuration du décodeur du fichier HEX dans la platine du décodeur, il faut utiliser un logiciel de programmation de PIC, branché sur le connecteur à 7 points, et un programmateur de PIC. Par exemple le **K150**. (photo ci dessous). On réinjecte le fichier HEX dans le programmateur. Le pic est programmé sur la platine du kit après qu'il soit soudé. Ensuite dans le logiciel de programmation (par exemple microBurn), ne pas oublier de sélectionner le mode "programmation sur platine cible", c'est à dire le menu "option" puis "**ICSP mode**". C'est un menu qui fonctionne en bascule. Le pic est un 16F684.

Il est possible que la vérification de la programmation échoue lors de la relecture de l'EEPROM. En effet, le pic a été configuré pour que l'octet d'adresse 10 soit verrouillé. Il contient la valeur de calibration (OSCCAL) du timer du PIC et ne peut normalement pas être changé. Il est possible d'utiliser le programmateur "MPLABX IPE".



DATA vers PDGD (P6)
 PCLK vers PGC (P5)
 VPP vers VPP (P4)
 0V vers GND (P1)
 5V vers VDD (P2)

Configuration « manuelle » du décodeur de signal digitalBahn (si on ne dispose pas de programmeur)

Pour configurer le décodeur, il faut appuyer sur le bouton poussoir, il se met alors en mode « apprentissage d'adresse ». La première led du feu sur lequel est connecté le décodeur se met à clignoter (c'est la première ligne du tableau ci-dessus). A l'aide du système de programmation numérique à la raquette, sélectionner le mode de pilotage d'aiguillage, puis sélectionner une adresse et envoyer un + pour valider la première adresse. Lorsque c'est fait, c'est la ligne suivante qui est validée. Il faut faire 14 fois la procédure. Il n'est pas nécessaire d'appuyer à chaque fois sur le bouton.

NB :

la commande + correspond au code « vert » du fichier HEX, soit 2 c'est à dire 10 en binaire
la commande – correspond au code « rouge » du fichier HEX, soit 1 c'est à dire 1 en binaire

Fichiers PIC pour le décodeur DigitalBahn

Fichier HEX décodeur pour feux SNCF version DCC modifié avec le clignotement des feux à 2Hz
au lieu de 1Hz, qui correspond à la réalité « decodeur.X.production.hex »
<http://cdmrail.free.fr/ForumCDR/download/file.php?id=6283>

Fichier HEX décodeur simple pilote de leds
http://www.digital-bahn.de/hex/led/led_16f684_schalten_355_dcc.hex

Codes commandes pour le décodeur digitalBahn

Platine avec composants, avec pic programmé pour signaux allemands 18€
Nécessite de reprogrammer le pic avec l'applicatif « decodeur.X.production.hex »
Il s'agit un décodeur à base de CMS.

https://digital-bahn.de/shopping/led_dekoder_002

Pour le choix du pays, choisir France et non pas France Métropolitaine.
Il s'agit d'un kit à composants CMS à souder soi même.

Logiciels PIC pouvant équiper le décodeur digital Bahn

Led_signal_10.hex

Logiciel d'origine d'exploitation des signaux complexes SNCF fourni par digital Bahn. Clignotement des feux à 1 Hz

decodeur.X.production.hex

Logiciel d'exploitation des signaux complexes SNCF modifié pour le clignotement des feux à 2 Hz.

Led_Schalten.hex

Logiciel d'exploitation pilotant les leds des feux à la demande. Utile pour piloter les leds des feux directionnels

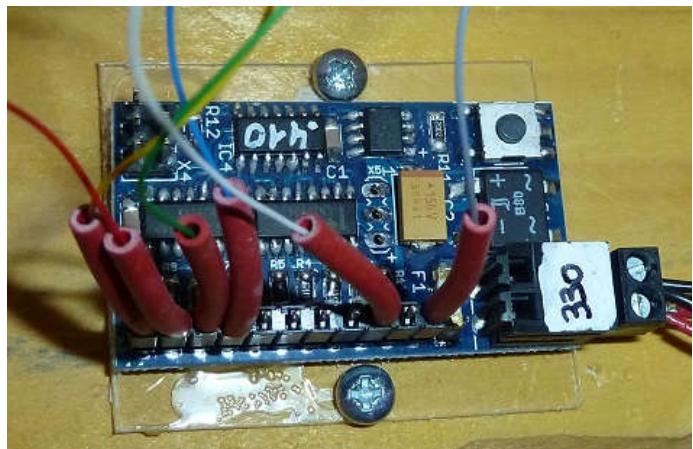
Il existe d'autres logiciels fournis par DigitalBahn sur leur site pour d'autres effets d'allumage qui peuvent être utilisés pour l'ambiance de l'éclairage d'un réseau (exemples : simulation allumage de néons, simulation de poste à souder, clignotement pour les feux de passage à niveau...). Tous ces fichiers HEX sont configurables avec le logiciel HEX_MANIPU, et utilisables dans le même décodeur.

L'injection d'un logiciel dans le pic du décodeur nécessite l'utilisation d'un programmeur de PIC.

Mise en place du décodeur DigitalBahn

Le décodeur digitalBahn étant sous la forme d'une platine sans trous de fixations, on peut la coller du ruban adhésif électrique sur un support plexiglas percé à deux endroits. Il est ainsi plus aisément de fixer le décodeur sous le réseau. La colle néoprène n'est pas définitive et il est toujours possible d'enlever le décodeur du support par décollement.

Un connecteur HE10 femelle a été inséré dans le support HE10 mâle existant. On peut ainsi souder les fils du feu sur le support et enlever le décodeur si nécessaire.



Attention toutefois, la colle néoprène et d'autres colles à phase solvantée attaque le cuivre au bout de quelques années et finit par dissoudre les pistes non protégées par le vernis.

Procédure pour piloter les signaux branchés sur les décodeurs DigitalBahn :

Envoi_signalBahn (adresse)

adresse : adresse du signal sur le bus DCC

Utilisation :

```
Envoi_signalBahn(476)
```

Utilisation du décodeur CDF80108007S pour les signaux complexes

Ce décodeur est utilisable dans différents modes. Seul le mode 0 est utilisable avec toutes les centrales. Les autres modes utilisent le protocole DCC étendu qui permet d'allumer deux sorties à 1 sur la même adresse, ce qui n'est pas compatible avec toutes les centrales. Cette combinaison de deux sorties à 1 sur la même adresse permet de rendre une led clignotante. Les feux jaune clignotant, vert ou rouge nécessitent donc 2 sorties et le câblage doit être adapté, ainsi que la routine de pilotage du programme complexe. Ne disposant pas de ce décodeur, aucun essai n'a pu être réalisé.

Mode 0 : adressage individuel des deux sorties d'une adresse : 0 0, 0 1, 1 0 mais pas 1 1

Mode 1 : commande de 4 signaux à 2 états

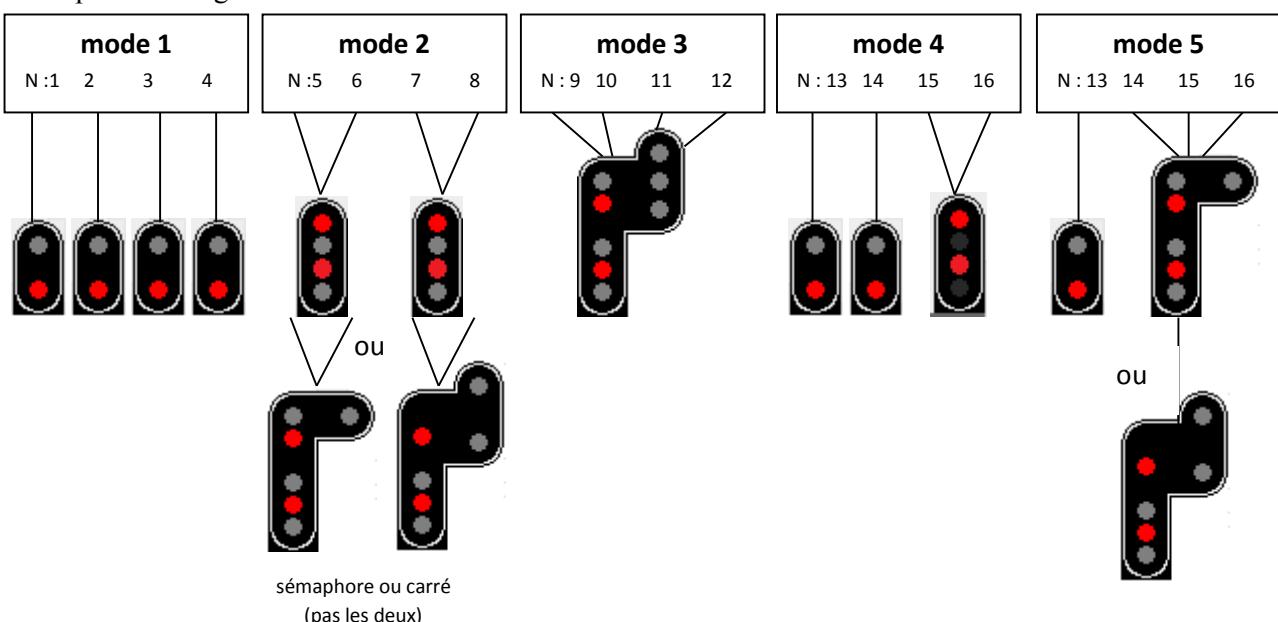
Mode 2 : commande de 2 signaux à 4 états

Mode 3 : commande d'un signal à 6 états (sans clignotement) ou 8 états (si clignotement) (8 feux)

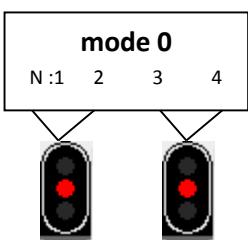
Mode 4 : commande de 2 signaux à 2 états, un signal 4 états

Mode 5 : commande d'un signal à 2 états, un signal à 5 états (sans clignotement) ou 6 états (si clignotement)

Exemple de configuration :



Le mode 0 peut être configuré pour gérer 2 signaux à 3 feux :



cette configuration ne permet pas de faire clignoter le feu jaune.

Les deux premières sorties de l'adresse N de base d'un signal sont en général affectées au carré et au sémaphore
Les deux sorties de l'adresse suivante (N+1) sont affectées en général au vert et à l'avertissement

Les deux sorties de l'adresse N+2 sont affectées en général au ralentissement/rappel et au rappel

Les deux sorties de l'adresse N+3 sont affectées en général au rappel de ralentissement.

Dans le programme client, la routine de pilotage des décodeurs CDF s'appelle *envoi_CDF*.

Procédure pour piloter les signaux branchés sur les décodeurs CDF :

Envoi_CDF(adresse)

adresse : adresse du signal sur le bus

Appel de la procédure pour piloter le signal à l'adresse 476 :

Utilisation : **Envoi_CDF(476)**

affiche un sémaphore sur le signal d'adresse de base 476 piloté par un décodeur CDF

Rappel à propos de la signalisation combinée :

code contient le motif de bits correspondant aux états à activer sur le signal. En cas de signalisation combinée, 2 bits sont à 1 (exemple avertissement et rappel 60). La fonction **code_to_aspect** renvoie le premier **aspect** du signal et le deuxième dans une variable globale « **Combine** ».

Exemple :

code=10100000000 (en binaire)

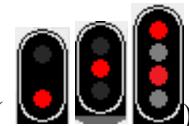
aspect=code_to_aspect(code)

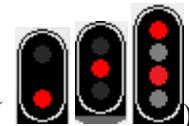
aspect=jaune (8) et Combine=ral_30 (10)

Utilisation du décodeur LDT-DEC-SNCF pour les signaux complexes

Ce décodeur peut intégrer de un à 4 signaux. L'adressage des signaux dépend donc du câblage.

Un décodeur occupe 8 adresses (partie haute du décodeur de N à N+3, partie basse=N+4 à N+7). Le cavalier J2 permet de sélectionner le protocole Motorola ou DCC. Le cavalier J1 permet de sélectionner des signaux avec un commun au + ou au -. Ne disposant pas de ce décodeur, aucun essai n'a pu être réalisé. Attention, ce décodeur ne permet que de piloter des feux alimentés en 5V. Chaque sortie est tamponnée par une résistance de 300 Ohms.

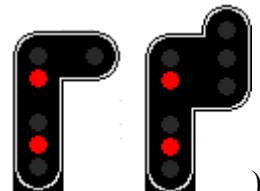


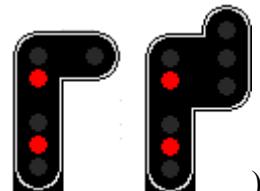
2 adresses permettent de piloter un signal à 2 feux, 3 feux ou à 4 feux. ()

Le pilotage se fera comme suit pour ces signaux (voir doc LDT)

Dans le programme client, on appellera ce mode le mode 1 :

adresse N, commande - (rouge) :	01	=	Sémaphore
adresse N, commande + (vert) :	10	=	voie libre
adresse N+1, commande - (rouge) :	01	=	carré (si utilisé)
adresse N+1, commande + (vert) :	10	=	Avertissement



3 ou 4 adresses permettent de piloter un signal à plus de 4 feux ( ,)

Le pilotage se fera comme suit pour ces signaux (voir doc LDT)

Dans le programme client, on appellera ce mode le mode 2 :

adresse N+2, commande - (rouge) 01 = **demande groupe 1**

puis

adresse N, commande - (rouge) :	01	=	Sémaphore	commandes du groupe 1
adresse N, commande + (vert) :	10	=	voie libre	
adresse N+1, commande - (rouge) :	01	=	carré	
adresse N+1, commande + (vert) :	10	=	Avertissement	

adresse N+2, commande + (vert) 10 = **demande groupe 2**

puis

adresse N, commande - (rouge) :	01	=	carré violet	commandes du groupe 2
adresse N, commande + (vert) :	10	=	blanc	
adresse N+1, commande - (rouge) :	01	=	sémaphore	
adresse N+1, commande + (vert) :	10	=	Aspect 8	

adresse N+3, commande - (rouge)	01	=	demande groupe 3
puis			
adresse N, commande – (rouge) :	01	=	Avertissement clignotant + Ralentissement 60
adresse N, commande + (vert) :	10	=	vert clignotant
adresse N+1, commande - (rouge) :	01	=	Disque D
adresse N+1, commande + (vert) :	10	=	Avertissement clignotant
adresse N+3, commande + (vert)	10	=	demande groupe 4
puis			
adresse N, commande – (rouge) :	01	=	Ralentissement 30
adresse N, commande + (vert) :	10	=	Ralentissement 60
adresse N+1, commande - (rouge) :	01	=	Rappel de ralentissement 30
adresse N+1, commande + (vert) :	10	=	Rappel de ralentissement 60

commandes du groupe 3

commandes du groupe 4

la présentation du rappel 30 ou 60 + avertissement ou avertissement clignotant n'est pas possible avec ce décodeur.

La routine à utiliser dans le programme client est **Envoi_LDT**

Appel de la procédure :

Envoi_LDT (adresse)

Adresse : adresse DCC sur le bus

Exemple :

Envoi_LDT (15)

Utilisation du décodeur LEB pour les signaux complexes

Ce décodeur utilise 8 adresses consécutives qui peut être programmé de façon standardisée ou spécifique. Cette présentation ne concerne que l'utilisation standardisée (**CV3=0 et CV5=0** : table d'allumage par défaut) qui est compatible avec toutes les versions logicielles du décodeur. Pour utiliser ce décodeur avec les signaux complexes, on peut se limiter à écrire 5 adresses.

L'affichage d'un signal complexe s'effectue en écrivant les 5 adresses d'accessoires du décodeur suivant une combinaison prédéfinie. Le tableau ci-dessous montre les combinaisons prédéfinies. Un 0 dans le tableau signifie que l'adresse correspondante est à l'état inactif, un 1 signifie que l'adresse correspondante est à l'état actif.

L'adresse **impaire** du décodeur Adr est à mettre dans les CV9 et CV1 suivant les formules ci-dessous:

CV9 = Adr div 64 (division entière)

CV1 = Adr mod 64 (reste de la division entière)

L'adresse d'accessoire Adracc qui sera visible par la centrale sur le bus DCC est donnée par la formule:

$$\text{Adracc} = 4 \times (\text{Adr}-1) + 1$$

Exemple : Si Adr = 125, alors CV9=1 ; CV1=6 et Adracc = 497

Le décodeur occupera les adresses Adracc à Adracc+7 soit 8 adresses sur le bus DCC, mais seules les 5 premières sont utilisées pour établir l'affichage du feu.

Code	Signal affiché	Adr+0	Adr+1	Adr+2	Adr+3	Adr+4
0	Carré	0	0	0	0	0
1	Carré violet	1	0	0	0	0
2	Blanc clignotant	0	1	0	0	0
3	Blanc	1	1	0	0	0
4	Sémaphore	0	0	1	0	0
5	Sémaphore clignotant	1	0	1	0	0
6	Rappel 30	0	1	1	0	0
7	Rappel 60	1	1	1	0	0
8	Avertissement	0	0	0	1	0
9	Ralentissement 30	1	0	0	1	0
10	Avertissement clignotant	0	1	0	1	0
11	Ralentissement 60	1	1	0	1	0
12	Vert clignotant	0	0	1	1	0
13	Vert	1	0	1	1	0
14	Rappel ralentissement 30 + Avertissement	0	1	1	1	0
15	Rappel ralentissement 30 + Avertissement clignotant	1	1	1	1	0
16	Rappel ralentissement 60 + Avertissement	0	0	0	0	1
17	Rappel ralentissement 60 + Avertissement clignotant	1	0	0	0	1
18	ralentissement 60 + Avertissement clignotant	0	1	0	0	1

Utilisation : Envoi_LEB(497)

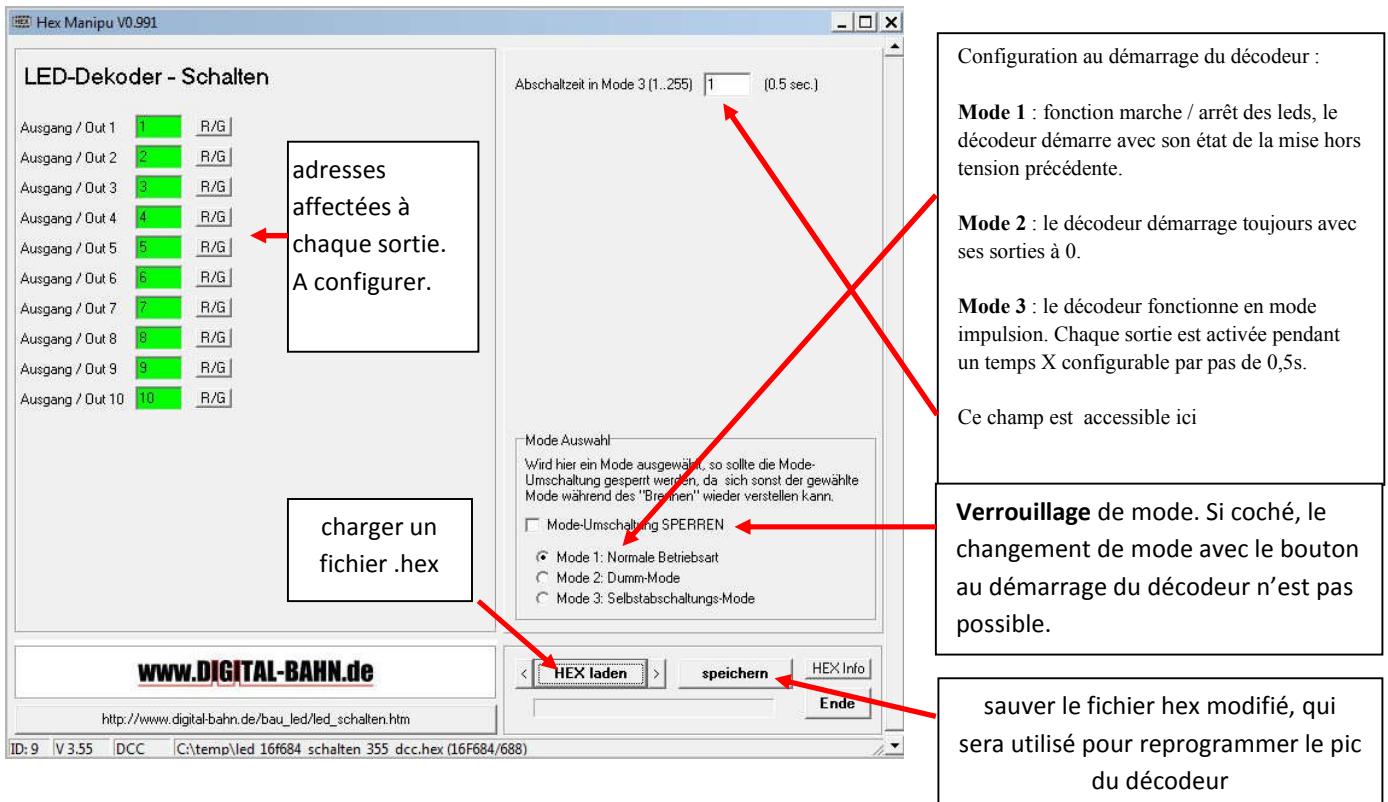
NB :

Le programme client envoie un 1 à l'adresse considérée pour passer l'adresse à l'état actif (1)

Le programme client envoie un 2 à l'adresse considérée pour passer l'adresse à l'état inactif (0)

Configuration du décodeur Digitalbahn pour les feux directionnels ou les TIV

Il faut équiper le PIC du décodeur et le paramétrer avec le logiciel « Schalten » avec le programme HEX_MANIPU. Le fichier HEX obtenu doit être réinjecté dans le PIC du décodeur avec un programmateur de PIC.



Le tableau ci dessus signifie que pour allumer la led de la sortie 1 au signal, il faut envoyer « 2 » à l'adresse DCC 1 car l'adresse est codée en vert. Si elle est codée en rouge, il faut envoyer « 1 ». Pour éteindre la led, il faut envoyer « 1 » à l'adresse DCC 1.

Ce pilotage est réalisé par le programme client par cette fonction :

```
Adresse:=1;
pilote_acc(adresse,2,feu); // sortie 1 à 1; allumage led 1
```

Dans tous les cas, un fichier HEX modifié doit être réinjecté dans le PIC du décodeur.

Voir la procédure « Configuration par programmateur », page 20.

Pour une configuration manuelle sans programmateur, voir « Configuration « manuelle » du décodeur de », page 22.

Le câblage d'un feu directionnel couplé à un signal complexe est montré en annexe 2.

Description logicielle du programme client SIGNAUX_COMPLEXES

Pour les programmeurs, signaux complexes GL a été écrit en Delphi7. Le code est compatible avec Rad Studio 12 (Delphi12) (Embarcadero et est compilable avec ces plateformes. Si on compile avec Rad studio, on ne peut plus revenir à Delphi7. Une directive de compilation permet de générer le code spécifique à D12 : {\$IF CompilerVersion>=28.0} : signifie que ce qui suit sera compilé pour un compilateur >=28 soit Delphi>=11.

Modification du programme avec Delphi 7

Delphi7 est téléchargeable gratuitement ici (après avoir créé un compte).

<https://delphi.developpez.com/telecharger-gratuit/delphi7-perso/>

la clé est la suivante

Numéro : YH?Z-2WDEGK-S48529-3AS3

Clé : G5N-D95

Le lancer la première fois en "Executer en tant qu'administrateur" (notamment pour W11).

Il sera peut-être nécessaire d'autoriser l'écriture du répertoire Programmes (x86) pour éviter de lancer systématiquement Delphi en mode admin. On peut aussi utiliser le mode de compatibilité "Windows XP" (clic droit en menu étendu / résoudre les problèmes de compatibilité). Nota. Pour Windows 11, il est conseillé de désactiver l'UAC (paramètres windows, chercher "UAC") et mettre le curseur à 0.

A la première exécution de Delphi, choisir "s'enregistrer ultérieurement".

Il reste à configurer Delphi pour y intégrer les composants additionnels Sockets et l'activex TmsComm:

Note : Après la version 9.1 de signaux_complexes compilée avec D12, les unités de AsyncProc ont été utilisées pour la communication COM/USB, rendant inutile le composant TMSCOMM (MsComm32.ocx). Ces unités présentent l'avantage de ne pas déclarer de composant supplémentaire, et d'être compilable en 32 ou 64 bits. Une directive de compilation a été ajoutée dans l'unit « UNITPRINC.PAS » pour décider de l'utilisation d'AsyncPro lors de la compilation avec D12. Dans ce cas, le composant TMSCComm devient inutile. AsyncPro n'est pas compilable avec Delphi 7, cette version nécessite le composant TMSCOMM.

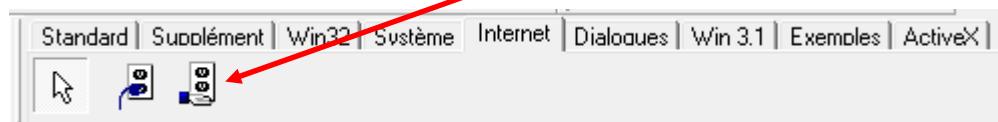
Installation des composants sockets (TClientSocket et TServerSocket) pour Delphi7

Télécharger le fichier **dclsockets70.bpl** depuis un site internet.

Menu composants / installer des paquets, cliquer sur ajouter et sélectionner "dclsockets70.bpl" et ouvrir. bouton ajouter un paquet de conception

naviguer dans c:\programmes\borland\Delphi7\bin
et choisir le fichier dclsockets70.bpl
cliquer sur OK

Les composants ClientSocket et ServeurSocket apparaissent dans l'onglet Internet



Installation de Rad Studio 13 (Delphi13)

<https://downloaddevtools.com/en/product/4064/free-download-embarcadero-rad-studio-delphi-13-florence>

Choisir Embarcadero RAD Studio Delphi13.0 Florence V37. ... (9.1 Go)

Désactiver l'antivirus, Windows defender, le pare feu et la sécurité des applications et la sécurité en temps réel. On peut essayer de créer une exception sur le fichier exe ci-dessous avant de les rétablir, mais il est possible que windows supprime le fichier.

Dézipper les fichiers internes (le mot de passe est DownloadDevTools.ir) et trouver

RADStudio-13-xxxxxx-KeyPatch.exe

Lancer le **keyPatch** en mode administrateur et suivre les instructions dans la fenêtre colorée.
(la 1ere étape est de télécharger

https://altd.embarcadero.com/download/radstudio/13.0/RADStudio_130_17a92.iso puis de l'extraire)

Lors du choix des caractéristiques d'installation, ne pas oublier de cocher l'aide à droite. DÉCOCHER "C+" à gauche.

Attention, il est possible que le nom de domaine embarcadero soit bloqué dans le fichier hosts:
C:\Windows\System32\drivers\etc\Hosts , on peut les supprimer dans un éditeur de texte.

Installation des composants sockets (TClientSocket et TServerSocket) pour RAD Studio

Attention : Les composants sockets ne sont disponibles que pour la version 32 bits de Delphi13. Il faut donc lancer D13 et pas D13 64 bits

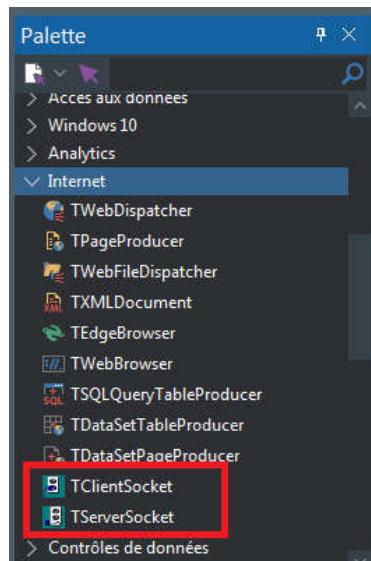
Menu composants / installer des packages, cliquer sur ajouter

Aller dans C:\Program Files (x86)\Embarcadero\Studio\37.0\bin

Choisir le fichier **dclsockets370.bpl** puis cliquer ouvrir, puis OK.

Le 37 peut varier en fonction de la version installée de RAD Studio.

Ouvrir ou créer un projet et cliquer sur composants pour vérifier que "TclientSocket et TserverSocket" s'y trouvent bien



Installation du composant MScomm32 pour Rad Studio

NB : le composant MSComm32 n'est pas nécessaire pour la compilation des versions D12 car la communication série utilise l'unité « AsyncPro » (Unités principales : AdPort et OoMisc).

Menu Composant / Importer un composant. (pas installer un composant)

Si vous avez installé C++, choisir "**VCL pour Delphi win32**"

Choisir importer un contrôle Active X puis cliquer suivant.

Vérifier dans la liste dans la colonne description la présence de "**Microsoft Comm Control 6.0**". Cliquer dessus. (S'il est absent de la liste, faire "Ajouter" et rechercher le fichier "mscomm32.ocx" fourni avec le zip de signaux_complexes)

Faire Suivant et encore Suivant.

Choisir "**Installer dans un nouveau package**", cliquer sur suivant

Appeler le **nom du package** activeX, et dans sa **description** également.

Cliquer Terminer.

A la question "l'unité MSCommLib_TBL.pas" référence le framework..... répondre **oui**

Tout fermer, inutile de sauvegarder

Le composant TMSCComm est dans l'arborescence de la palette :

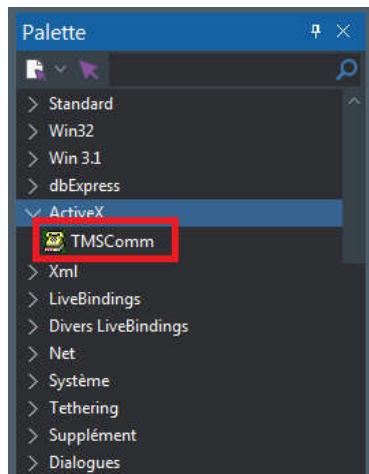
Option :

Choisir VCL pour Delphi win32

Choisir active X

Choisir ajouter et chercher le fichier mscomm32.ocx. S'il a déjà été ajouté, il est présent dans la liste à la rubrique Microsoft Comm control 6.0.

Sélectionner la ligne Microsoft Comm control 6.0



Personnalisation RadStudio12:

Compilateur:

Projet / options / Construction / Compilateur Delphi / compilation

Décocher vérification de débordement / Vérification des limites / Vérifications des E/S

Dévalider l'auto complétion et la ligne en cours en surbrillance

Menu Outils / Options / Editeur / Langage

Sélectionner Langage : "**Delphi**"

Onglet options : décocher "ligne en cours de surbrillance"

Onglet "**Audit de code**" : décocher :

- **Achèvement de template de code.**
- **Achèvement de bloc.**
- **Achèvement de code.**

Mettre l'éditeur en mode sombre :

Menu outils / options

Menu Editeur / Couleur

Couleurs prédéfinies : choisir **Crépuscule**.

Cliquer **enregistrer**

Si l'aide est absente :

Menu outils / Gérer les fonctionnalités

Cliquer sur Aide à droite.

Barre d'outils des composants en haut :

Menu Voir / Barres d'outils, cocher: composants

A la fenêtre de démarrage si une erreur sur les packages getit en erreur s'affiche, il faut aller dans l'éditeur de registres windows.

Aller à : HKEY_CURRENT_USER\Software\Embarcadero\BDS\23.0\CatalogRepository

Supprimer les **valeurs** des clés **DeferredComponents** et **DeferredComponentsFailed**

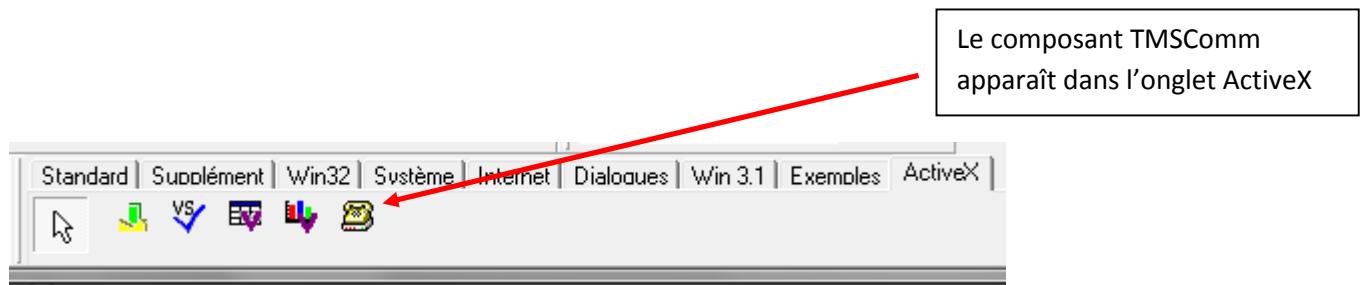
Debuggeur

Outils / options / options du débogueur (liste de gauche) / debogueur Embarcadero / Exceptions du langage:
Décocher "notifier sur les exceptions du langage".

Installation du composant MScomm32 pour Delphi7

Menu composants / importer un contrôle activeX / cliquer sur ajouter
chercher le fichier mscomm32.ocx (qui est dans le répertoire signaux_complexes_gl)
cliquer sur ok

Dans la liste, sélectionner « Microsoft Comm Control 6.0 (Version 1.1)
Cliquer sur installer puis sur OK. Il est inutile de sauvegarder l'unité créée.



Delphi7 :

Compilateur :

Projet / options / Découcher vérification de débordement / Vérification des limites / Vérifications des E/S

Débugger:

Dans outils / options du déboggeur / onglet Exceptions du langage, **décocher** « Arrêter sur exceptions Delphi »

Défaillances possibles sous DELPHI7

En cas de « défaillance ouverture fiche à l'ouverture du source du programme » sous Delphi:

Fermer Delphi7

Editer en ascii le fichier UnitPrinc.dfm

Supprimer les lignes concernant le composant MSCommRelais:TMSComm (voir ci dessous)

Sauvegarder

Réouvrir Delphi7 et le projet Signaux_complexes_gl.dpr

1. faire glisser le composant TMScomm dans la fiche UnitPrinc

Si une erreur apparait (Tmscomm existe déjà), il faut supprimer la ligne de déclaration du composant TMSCOM dans le fichier (UnitPrinc) (MSComm1: TMSCOM;)

2. renommer le composant en MSComm1

3. Affecter la procédure d'interruption MSComm1Comm dans le champ onComm du composant MSComm.(y mettre MSComm1Comm)

Fini

Lignes à supprimer dans UnitPrinc.dfm:

```
object MSCommxxx: TMSComm
  Left = 440
  Top = 86
  Width = 32
  Height = 32
  OnComm = MSCommUSBLenz
  ControlData = {
    2143341208000000ED030000ED0300001568A640000600000010000040000
    0002000080250000000008000000000000000000003F00000011000000}
end
```

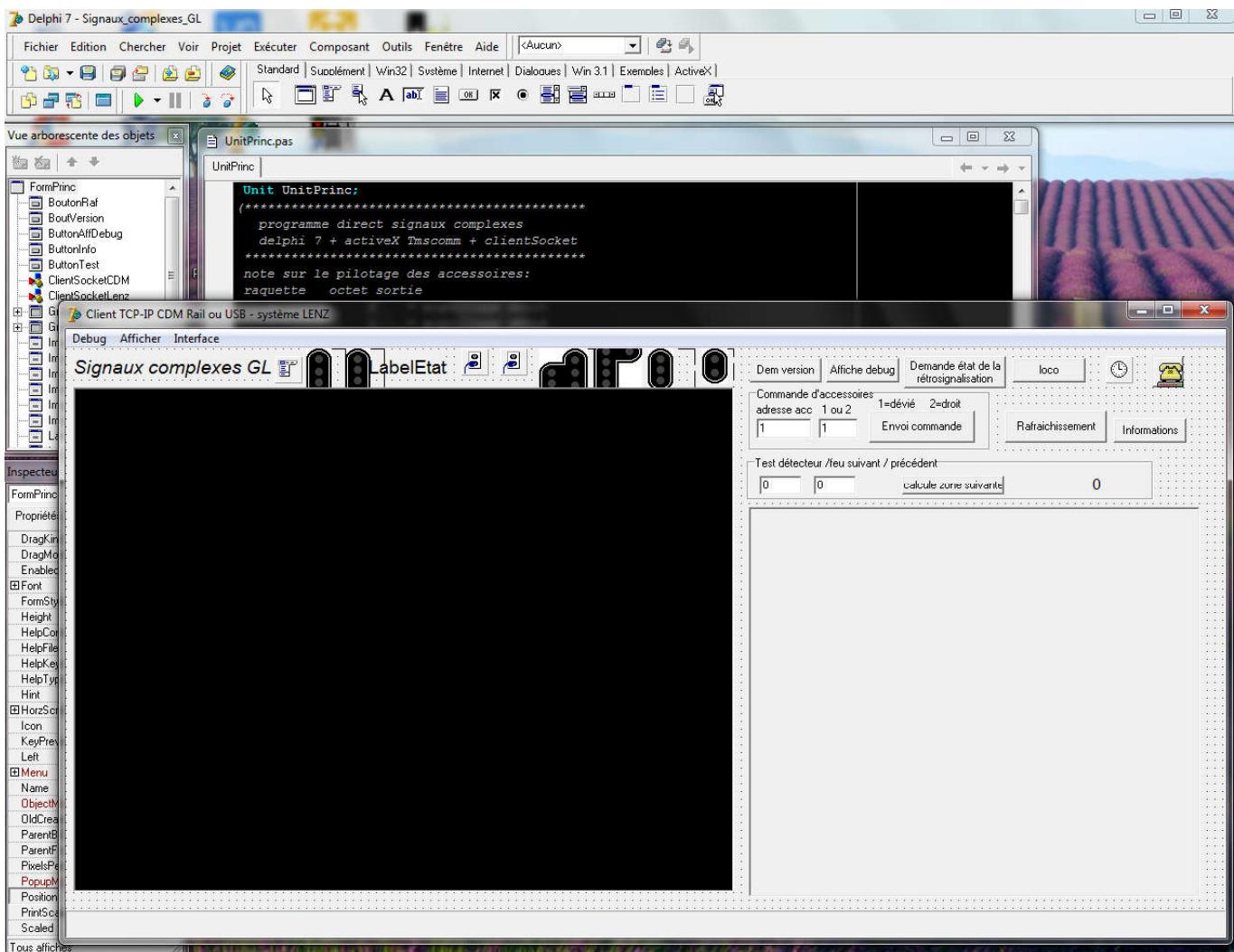
En cas d'exception « classe non enregistrée » à l'exécution de signaux_complexes_GL :
Il faut enregistrer mscomm32.ocx dans le registre avec la commande : (ouvrir une console en mode admin)

```
regssvr32 mscomm32.ocx
```

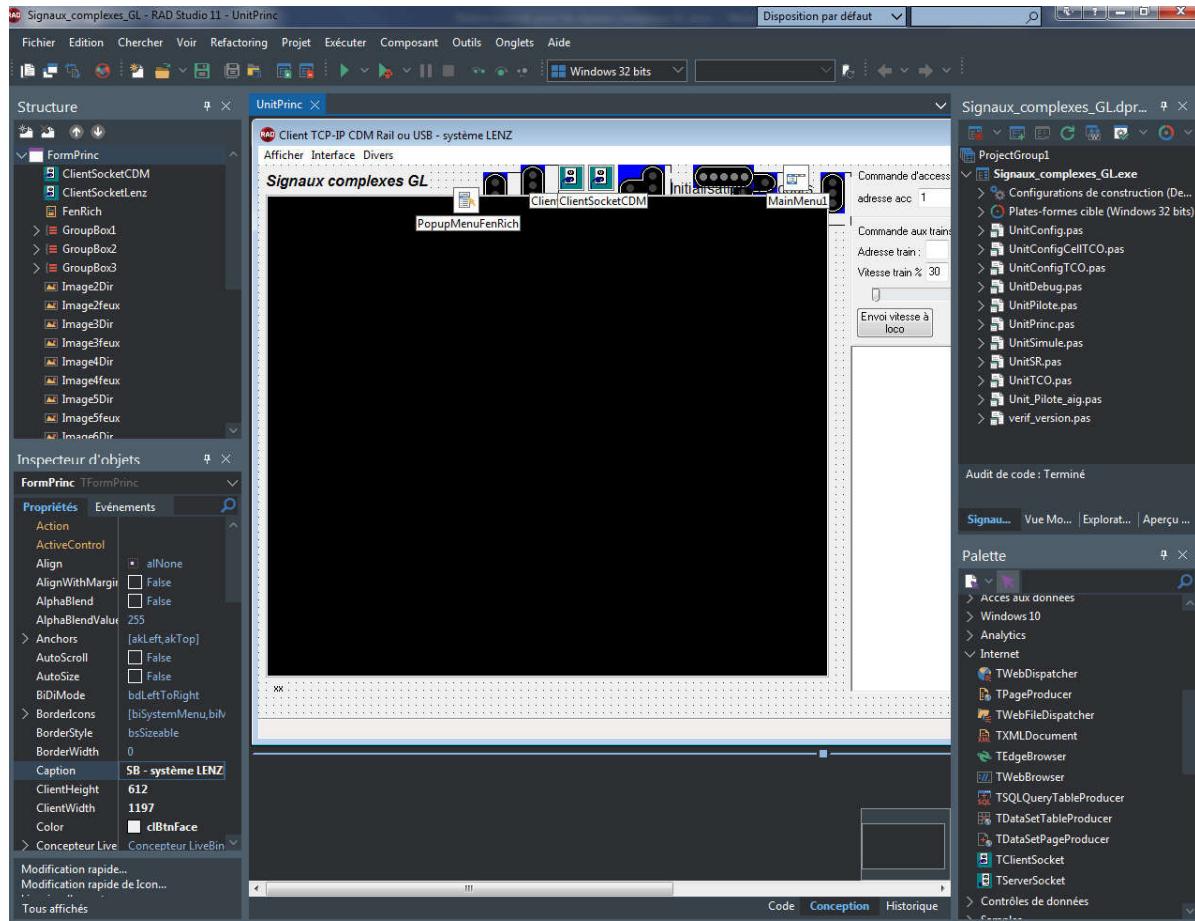
En cas de message « les informations de licence Borland ont été trouvées mais elles ne sont pas valides »

Aller dans le répertoire c:\utilisateurs\ votre_session\borland
Supprimer le fichier registry.slm
Relancer delphi, il va demander l'enregistrement et recréer un fichier valide.

L'environnement de travail sous Delphi une fois tout installé, avec le projet ouvert :



L'environnement de travail avec Rad Studio:



Le nom du projet avec Rad Studio est Signaux_complexes_GL.dproj

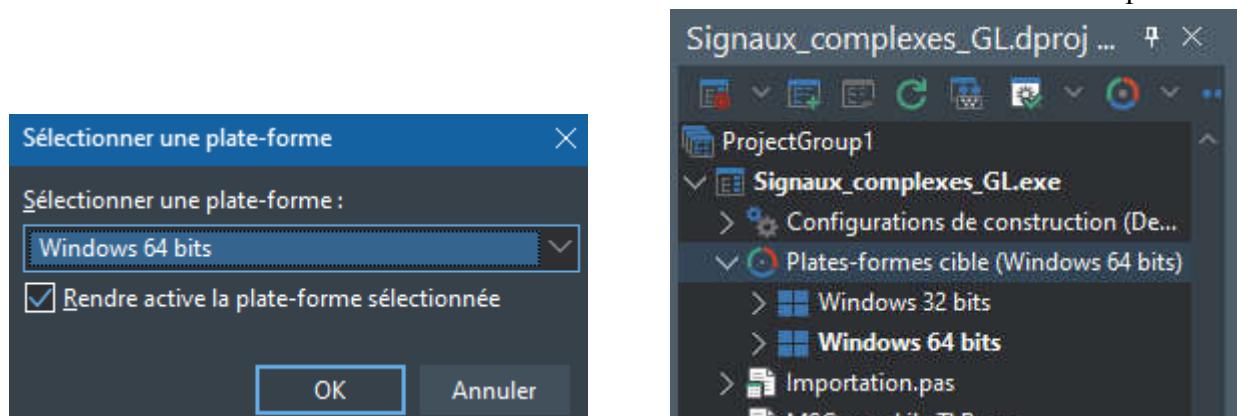
Compilation en 32 ou 64 bits avec RadStudio 12 :

Ouvrir le gestionnaire de projet :

Voir / Fenêtre d'outils / Projets.

Dans la fenêtre de projet, faire un clic droit sur « Plates-formes cibles (Windows 32) » et cliquer sur « ajouter une plate forme »

Une fenêtre s'ouvre et sélectionner Windows64 bits et laisser cocher « rendre active la plateforme ».



La plateforme Windows 64 bits est active (en gras)

On peut alors construire le projet en 64 bits. Seule la compilation avec ASyncPro fonctionnera.

Généralités sur la détermination des routes sur les évènements détecteurs

En mode connecté à la centrale (mode autonome), la rétro-signnalisation envoie les évènements des détecteurs dans la procédure `decode_retro` qui isole, entre autres, les évènements détecteurs. Celle-ci remplit le tableau `event_det`. Si un front descendant est détecté sur un détecteur, elle appelle la procédure `calcul_zones` qui traite le tampon.

En mode connecté à CDM rail, on récupère les messages émis par le serveur de CDM via le socket dans la procédure `TFormPrinc.ClientSocketCDMRead`. Celle-ci remplit le tableau `event_det`. Si un front descendant est détecté sur un détecteur, elle appelle la procédure `calcul_zones` qui traite le tampon.

Le tableau `event_det` contient un nombre limité d'évènements détecteurs, qui se dépile lorsqu'ils sont traités.

La procédure `calcul_zones` met en forme le tampon `event_det`. Lorsqu'au moins 2 détecteurs différents sont présents dans le tampon, la procédure tente de relier les deux détecteurs afin de vérifier s'ils ont été activés par la même locomotive. Pour cela, les détecteurs doivent être adjacents, et séparés par des aiguillages dont la route est tracée entre les deux détecteurs. C'est le rôle de la fonction `calcul_zones_det(det1, det2)` qui renvoie 10 si c'est le cas et on déclare qu'une route est validée, puis on supprime le premier détecteur du tampon. La procédure `calcule_zones` balaye ensuite le reste du tampon `event_det` pour trouver d'autres routes entre d'autres détecteurs.

La fonction `calcul_zones_det(det1, det2)` vérifie que les deux détecteurs sont contigus, et s'ils sont séparés par des aiguillages, si l'itinéraire correct sur ces aiguillages est tracé entre les deux détecteurs. Afin de valider la route, la procédure teste le détecteur précédent avant `det1`.

La fonction retourne également le détecteur suivant à atteindre par la locomotive. Dans le cas d'une route trouvée, le résultat de la fonction est 10, et la mémoire précédente `event_det_tick[i]` n'a pas été traitée, et `MemZone(det2, DetSuivant)=TRUE`.

Dans la procédure `calcul_zones`, si une route a été trouvée, les feux sont mis à jour par la procédure `rafraichit` qui appelle `Maj_Feux`, qui met à jour les feux un par un par la procédure `Maj_feu`. C'est dans cette procédure `Maj_feu` que sont appliqués les organigrammes standardisés de pilotage des signaux complexes.

Voici un exemple d'activation des détecteurs activés par deux trains différents

```
513 à 531 => Mem 531 à 518
531 à 518 => Mem 518 à 523
526 à 513 => Mem 513 à 531
518 à 523 => Mem 523 à 526
513 à 531 => Mem 531 à 518
531 à 518 => Mem 518 à 523
523 à 526 => Mem 526 à 513
518 à 523 => Mem 523 à 526
```

premier train : route de 531 à 518
 Le détecteur 518 est le dernier de la ligne précédente, c'est donc le même train, le 1.
 Le détecteur 513 n'avait pas été engagé dans les lignes précédentes, c'est donc un nouveau train (2)
 le détecteur 523 était le dernier de la ligne du train 1, c'est donc le train 1.
 etc

L'algorithme détermine ensuite combien de train(s) sont engagés par détermination du détecteur terminal de la ligne précédente par rapport au détecteur initial de la ligne suivante.

```
train 1 513 à 531 => Mem 531 à 518
train 1 531 à 518 => Mem 518 à 523
train 2 526 à 513 => Mem 513 à 531
train 1 518 à 523 => Mem 523 à 526
train 2 513 à 531 => Mem 531 à 518
train 2 531 à 518 => Mem 518 à 523
train 1 523 à 526 => Mem 526 à 513
train 2 518 à 523 => Mem 523 à 526
```

Ces structures sont stockées dans des tableaux d'enregistrement :

```
TEvent_det_train : event_det_train : array[1..Max_Trains] of
    record
        NbEl : integer;      nombre de détecteurs enregistrés pour ce train (0 à 3)
        Det : array[1..3] of integer;
    end;
```

Exemple :

```
Event_det_Train[1].Det[1].det=531
Event_det_Train[1].Det[2].det=518
Event_det_Train[1].Det[3].det=523

Event_det_Train[2].Det[1].det=513
Event_det_Train[2].Det[2].det=531
Event_det_Train[2].Det[3].det=518
```

Le tableau du train1 signifie qu'il se déplace de 531 à 518, puis de 518 à 523 etc. On ne stocke que 3 détecteurs par train.

Ces informations sont affichées dans la fenêtre de debug dans le cadre en haut à droite.

Connexion aux interfaces en XpressNet en port COM ou USB

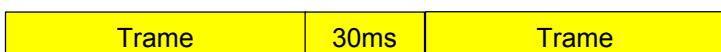
Les centrales Lenz nécessitent une temporisation entre l'envoi de 2 trames. Elle est fixée dans le programme à 30 ms. Ceci est géré dans la procédure `envoi_ss_ack(s : string);`

Protocoles 0 avec temporisation, 1 (xon xoff)



Les octets sont envoyés séparés d'une temporisation correspondant au paramètre "temporisation d'envoi des octets de la trame". Lorsque le dernier octet est envoyé, une temporisation de 30 ms est activée avant la trame suivante.

Protocoles 0 sans temporisation, 2 (rts-cts) , 4 (cts)



Les octets sont envoyés immédiatement sans temporisation. Lorsque le dernier octet est envoyé, une temporisation de 30 ms est activée avant la trame suivante.

Gestion de la liaison XpressNet ou DCC++ (mode autonome)

En mode autonome, la centrale transmet des informations via le bus XpressNet ou DCCpp de l'interface vers le PC soit par le port USB soit par le socket Ethernet.

La gestion des trames reçues de la centrale est effectuée dans la procédure `interprete_reponse`.

En fonction du type de trame reçue, le programme effectue différentes tâches.

Si une trame de rétrosignalisation est reçue, elle est traitée dans la procédure `decode_chaine_retro`.

La procédure `decode_chaine_retro` fait une première analyse de la trame. S'il ne s'agit pas d'une trame de service, une analyse spécifique est faite par la procédure `decode_retro`. Cette procédure est très importante car elle isole les événements détecteurs et aiguillages transmis par la centrale. Elle remplit le tampon `Event_det` pour les événements détecteurs, et met à jour le tampon de position des aiguillages sur les événements aiguillages.

La procédure `pilote_acc` permet de piloter les accessoires soit par CDM soit par la centrale.

```
procedure pilote_acc(adresse : integer; octet : byte; Acc : TAccessoire);
```

Adresse = adresse de l'accessoire

Octet = valeur de l'octet à envoyer à l'adresse de l'accessoire

Acc = Feu ou AigP

Feu permet de ne pas transmettre de temporisation d'attente avant d'envoyer un 0 à l'accessoire.

AigP attend une temporisation prédéfinie dans le fichier de configuration pour les aiguillages avant d'envoyer un 0. Cela permet à l'aiguillage de se positionner.

Les trames pour piloter les signaux et les aiguillages sont formatées ; elles sont envoyées à la centrale par la procédure `envoi`. Cette procédure envoie la trame par USB ou par le socket ethernet. Elle attend l'accusé de réception de la centrale.

Gestion du calcul de progression des trains (routes)

Une route permet déclarer qu'un trajet ou itinéraire a lieu entre deux détecteurs, en fonction de la position des aiguillages les séparant.

Des fonctions primitives servent au programme. Dans le programme, un « élément » désigne soit un détecteur ou un aiguillage. Ils sont identifiés par leur adresse et le type.

La fonction primitive la plus importante est la suivante :

```
function suivant_alg3( prec : integer ; typeELprec: integer;
                      actuel : integer ; typeELactuel: integer;
                      alg : integer) : integer;
```

Cette fonction renvoie l'élément suivant des deux éléments passés en paramètre (aiguillage ou détecteur) mais contigus. Elle renvoie également en variable globale : `TypeGen` le type de l'élément suivant.

Si les deux éléments `prec` et `actuel` ne sont pas contigus, on aura un résultat erroné.

Alg3 :

bit0 (1)=arrêt sur suivant qu'il soit un détecteur ou un aiguillage

bit1 (2)=arrêt sur aiguillage en talon mal positionné

bit2 (4)=arrêt sur aiguillage réservé

bits1 et 2: (2+4)=6= arrêt sur aiguillage en talon mal positionnée ou aiguillage réservé

bit3 (8)=arrêt sur un aiguillage pris en pointe dévié et `AdrDevie` contient l'adresse de l'aiguillage dévié ainsi que `typeGen`

code de sortie : élément suivant ou

9999 =erreur fatale

9998 = arrêt sur aiguillage en talon ou tjd/s mal positionnée

9997 = arrêt sur aiguillage dévié

9996 = arrêt sur position d'aiguillage inconnue

9995: arrêt anormal sur buttoir

9994: arrêt sur aiguillage réservé

TypeGen=1 si le résultat de la fonction concerne un détecteur, 2 s'il s'agit d'un aiguillage, 3 s'il s'agit d'un aiguillage BIS.

Exemple : suivant_alg3(518,1,20,2,1) ;

Renvoie l'élément de du prochain aiguillage pris en pointe dévié qui suit les éléments contigus « 518 » (détecteur) et « A20 » (aiguillage non BIS) . La variable globale TypeGen contiendra le type de l'élément.

```
// renvoie l'adresse du détecteur suivant des deux éléments contigus
function detecteur_suivant(prec : integer; TypeElPrec: integer;
                           actuel : integer; TypeElActuel: integer) : integer ;
```

Exemple : detecteur_suivant(518,1,25,2)

Renvoie le détecteur suivant de l'élément 518 (détecteur=1) à l'élément A25 (aiguillage=2) contigus.

Renvoie l'adresse du détecteur suivant des deux éléments

Det1 et Det2 peuvent être séparés par des aiguillages ou d'autres détecteurs (non contigus). Si les aiguillages sont mal positionnés entre det1 et det2, la fonction renvoie 0

```
function detecteur_suivant_El(el1: integer; TypeDet1: integer;
                               el2: integer; TypeDet2: integer) : integer ;
```

Renvoie les adresses des détecteurs adjacents au détecteur "adresse"

résultat dans adj1 et adj2

```
procedure Det_Adj(adresse : integer);
```

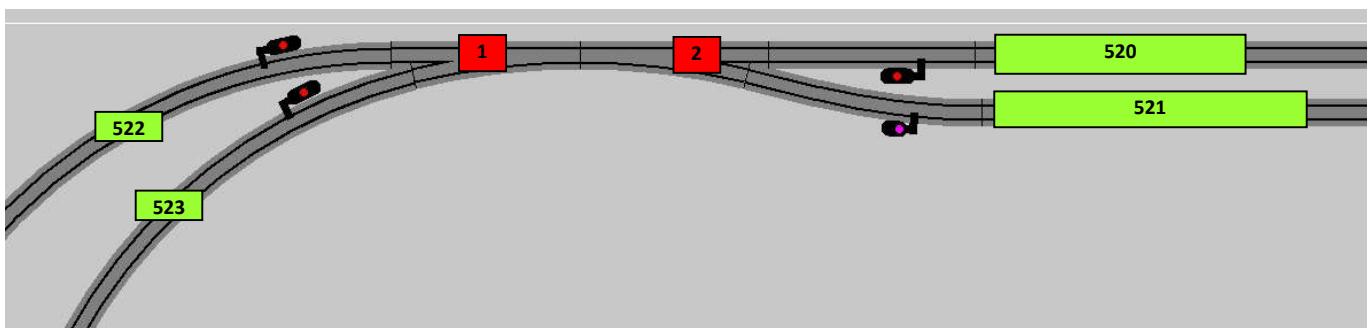
La procédure det_continu considère les deux détecteurs contigus det1 et det2.

S'ils sont séparés par des aiguillages, la procédure renvoie le dernier aiguillage précédent det2.

Si det1 et det2 ne sont pas séparés par des aiguillages, la procédure renvoie det1.

```
// renvoie l'élément avant det2 si det1 et det2 sont contigus ou séparés que par des
aiguillages
```

```
procedure det_contigu(det1,det2 : integer;var suivant : integer;var ElSuiv : TEquipement);
```



Exemples avec la situation ci-dessus:

Det_continu(522,520,suivant,ElSuiv)

Renvoie :

Suivant=2 ElSuiv=aig

`Det_continu(520,522,suivant,ElSuiv)`

Renvoie :

`Suivant=1 ElSuiv=aig`

`Suiv_alg3(522,det,1,aig,1)`

Renvoie 2 et TypeGen=aig

`Suiv_alg3(2,aig,1,aig,1)`

si l'aiguillage 1 est droit,

renvoie 522 et TypeGen=det

Récupération des informations sur la position des aiguillages

En mode CDM, les positions sont récupérées par la liaison client serveur IP. CDM envoie des positions codifiés de façon différente des centrales. Les informations sont reformatées comme suit:

Codification de la position des aiguillages

Transmis par CDM

Aiguillage simple

Dévié à gauche = 3

Dévié à droite = 1

Non dévié = 0

Transmis par le bus XpressNet

Dévié = 1

Droit (non dévié) = 2

Aiguillage triple

Dévié à gauche = 3

Aig1=1 et Aig2=2

Dévié à droite = 2

Aig1=2 et Aig2=1

Non dévié = 0

Aig1=2 et Aig2=2

TJD et TJS à 2 adresses (4 états)

Pos 1

Aig1=1 et Aig2=2

Pos 4

Aig1=1 et Aig2=1

Pos 5

Aig1=2 et Aig2=1

Pos 0

Aig1=2 et Aig2=2

TJD et TJS à 1 adresses (2 états)

Droite (en X) : Pos 0

2

Déviée ; Pos 1

1

La procédure `envoi_signal(int adresse)` aiguille le programme vers le pilotage des décodeurs correspondant aux signaux. Le pilotage proprement dit des signaux via les décodeurs est effectué dans les procédures de chaque décodeur.

Informations sur les aiguillages

`index:=index_aig(adresse) ; renvoie l'index de l'aiguillage "adresse"`

`aiguillage[index].position = const_droit = 2 : aiguillage non dévié`

`aiguillage[index].position = const_devie = 1 : aiguillage dévié`

Informations et procédures relatives aux signaux

`Maj_Etat_Signal(adresse,etat) : Procédure qui met à jour l'état du signal en tenant compte des règles de l'affichage en signalisation combinée (transforme l'état numérique « aspect » en code binaire).`
 Exemple : pour demander l'affichage d'un avertissement puis d'un rappel 60 sur le signal 316 :

```
Maj_Etat_Signal(316, jaune) ;
Maj_Etat_Signal(316, rappel_60) ;
```

à la fin de l'exécution des deux procédures, feux[index].EtatSignal=10000100000000 en binaire.
A la suite de ces 2 appels, si on lance

Maj_Etat_Signal(316, ral_30), feux[index].EtatSignal=100000000000 en binaire
(effacement des bits précédents car un ralentissement 30 efface l'avertissement). Ce motifs de bits sera alors utilisé dans la procédure de pilotage des décodeurs et allumera les feux en conséquence.

feux[index].EtatSignal : contient l'état du signal complexe ou du panneau directionnel codé en binaire ; défini dans l'unité UnitPilote :

```
bit 0 =carré
bit 1 =sémaphore
bit 2 =sémaphore clignotant
bit 3 =vert
bit 4 =vert clignotant
bit 5 =violet
bit 6 =blanc
bit 7 =blanc clignotant
bit 8 =jaune
bit 9 =jaune clignotant
bit 10 = ralentissement 30
bit 11 = ralentissement 60
bit 12 = rappel 30
bit 13 = rappel 60
bit 14 = Disque D (Décodeur LDT seulement)
bit 15 = ralentissement 60 + jaune clignotant (Décodeur LDT seulement)
bit 16 = aspect 8 (Décodeur LDT seulement)
```

NB : il ne peut y avoir au maximum que 2 bits à 1 simultanément dans Etat_SignalCplx. Exemple : le bit 8 et le bit 11 (cas d'une signalisation combinée).

```
// signal belge
// base
vert_jaune_H    =0 ; vert_jaune_H_F=1;
rouge          =1 ; rouge_F=2;
vertB          =2 ; vertB_F=4;
vert_jaune_V    =3 ; vert_jaune_V_F=8;
rouge_blanc    =4 ; rouge_blanc_F=16;
deux_jaunes    =5 ; deux_jaunes_F=32;
// combinée
chiffre        =6; chiffre_F=64;
chevron         =7; chevron_F=128;
clignote        =8; clignote_F=256;
Bita1           =15; Bita1_F=32768; // mise à 1 de bit signal combiné pour belge
```

Test_Signal(218,carre) (booléen) retourne TRUE si le signal « 218 » est au carré

feux[index].AncienEtat: contient l'état précédent du signal complexe avant changement d'état

Pour changer l'état d'un signal

Le programme principal n'utilise que 2 procédures:

Maj_Etat_Signal(adr,motif) // modification de l'état du signal

```
// exemple : Maj_Etat_Signal(AdrFeu,blanc)
```

Puis suivi, à la fin de la modification des états des signaux:

```
Envoi_signauxcplx;           // pilote les décodeurs de tous les signaux, dessine les états dans
                             // la fenêtre graphique et dans le TCO.
```

Information sur les détecteurs

`detecteur[adresse].etat = TRUE ou FALSE` : Contient l'état activé ou désactivé du détecteur (attention pas de l'actionneur)

`detecteur[adresse].train=chaine` contient le nom du train ayant actionné le détecteur d'adresse « adresse »

`MemZone [detecteur1,detecteur2]` = état de la zone de détecteur1 à détecteur2 (=TRUE occupée ; = FALSE libre) suivant le sens de circulation du train.

Procédures et fonctions

Renvoie l'adresse du détecteur suivant des deux éléments

Det1 et Det2 peuvent être séparés par des aiguillages jusqu'à 20 éléments maxi en sortie : 1= det1 non trouvé 2= det2 non trouvé ou code erreur>=9997 ou 0

```
function detecteur_suivant_El(el1: integer ; TypeDet1 : integer;
                               el2 : integer ; TypeDet2 : integer) : integer ;
```

Renvoie l'adresse du détecteur suivant des deux éléments contigus

```
TypeElprec/actuel: 1= détecteur 2= aiguillage 3=bis 4=Buttoir
function detecteur_suivant(prec : integer; TypeElPrec : integer;
                           actuel : integer; TypeElActuel : integer) : integer ;
```

Renvoie l'adresse de l'aiguille si elle est déviée après le signal et ce jusqu'au prochain signal sinon renvoie 0

adresse=adresse du signal
`function Aiguille_deviee(adresse : integer) : integer ;`

Renvoie vrai si les aiguillages au-delà du signal sont mal positionnés

```
function carre_signal(adresse : integer) : boolean ;
```

Renvoie vrai si une mémoire de zone est occupée du signal « adresse » au signal suivant

```
function test_memoire_zones(adresse : integer) : boolean ;
```

Renvoie l'état du signal suivant

si renvoie 0, pas trouvé de signal suivant.

rang=1 pour feu suivant, 2 pour feu suivant le 1, etc

dans AdresseFeuSuivant : adresse du feu suivant (variable globale)

```
function etat_signal_suivant(adresse,rang : integer) : integer ;
```

Calcul des zones depuis le tableau des fronts descendants des évènements détecteurs les détecteurs doivent être consécutifs

trouve le détecteur suivant de det1 à det2 si la route est correcte. (détecteurs en entrée obligatoires) transmis dans le tableau Event_det

Variable globale: El_suivant : adresse du détecteur suivant le détecteur "actuel"

Actuel,Suivant : nouveaux détecteurs du canton suivant

Résultat:

```
si 0 : pas de route
si 1 : détecteur det1 non trouvé
si 2 : détecteur det2 non trouvé
si 3 : erreur fatale
si 10 : ok route trouvée
function calcul_zones_det(det1,det2 : integer) : integer;
```

Donne le suivant au point de connexion de l'aiguillage. Cette procédure est interne à det_contigu.

```
prec=det ou aig ; suiv=aig
aig_suiv(527,7) : renvoie 520 dans suiv_2
procedure aig_suiv(prec,suiv : integer) ;
Procédure récursive.
```

```
Renvoie l'élément avant det2 si det1 et det2 sont contigus ou ne sont séparés que par des aiguillages
si det1 et det2 sont contigus sans aiguillages entre eux, ça renvoie det1
det_contigu(527,520) : renvoie 7 dans suivant
det_contigu(514,522) : renvoie 514 dans suivant
procedure det_contigu(det1,det2 : integer;var suivant : integer;var ElSuiv : TEquipement) ;
```

Chronologies d'appels du pilotage des trains en mode autonome

Calcul_zones_V1

```
1 det à 0 : Maj_route(det3)
1 det à 1 : pilote_train(det1,det3) ; maj_route(det3)
2 det à 0 : Maj_route(det3)
2 det à 1 : pilote_train(det2,det3) ; maj_route(det3)
0 det à 0 : maj_route(det3) ; pilote_train
0 det à 1 : pilote_train si pas de route affectée
```

Demarre_train

```
Aig_canton
Demarre_indes_train // roulage
```

Maj_route : appelé sur front descendant détecteur

```
Aig_canton(index_train,detecteur)
```

Aig_canton(idTrain, detect)

Réserve et arme la tempo d'arrêt des trainssi on arrive en fin de route
Si arrêt détecteur temporisé

Pilote_train : appelé sur front montant détecteur

Pilote les trains par rapport au signal

Train_sarrette

Si la route du train est terminée : Supprime_train(idtrain)
Roulage des trains



signal de ralentissement sur un réseau piloté par CDM Rail + programme client. Il annonce un aiguillage distant pris en pointe dévié à franchir à 30 km/h.
Ce signal sera suivi d'un signal de rappel de ralentissement 30 km/h (deux feux jaunes verticaux) placé avant l'aiguille.



panneaux directionnels sur signaux complexes annonçant la direction que prendra le train.

Fonctionnement de l'échange de données entre le client et le serveur CDM Rail

Lors de l'établissement de la liaison par le client au serveur (CDM), le type d'informations transmises au programme client est géré par une demande de services. Avec cette fonction, le client demande au serveur (CDM) le type de données (services) que le client souhaite disposer.

Les éléments d'informations nécessaires au fonctionnement du client signaux_complexes avec CDM rail sont les suivants:

- état des aiguillages
- état des actionneurs
- état des détecteurs
- position des trains

Les informations sont transmises de CDM au programme client à la seule initiative du serveur (CDM) sur un changement d'état. Par exemple ; on ne peut connaître l'état d'un aiguillage que s'il change de position. Lorsqu'il change de position, CDM (serveur) envoie au client son nouvel état. C'est le programme client qui mémorise l'état sur un changement reçu du serveur. Lors du démarrage du programme client, celui-ci ne connaît pas les états des éléments et sont positionnés par défaut. C'est lors de la phase de positionnement initial des aiguillages en début de RUN que les informations sont transmises au client.

Structure du fichier ConfigGenerale.cfg

La modélisation du réseau est contenue dans le fichier ConfigGenerale.cfg. Il s'agit d'un fichier ascii éditabile. La description du réseau est réalisée depuis le panneau de configuration. La description ci-dessous et dans les paragraphes suivants n'est donnée que pour information. Les variables qu'il contient peuvent également être modifiés depuis la fenêtre de configuration. Le nom des variables peut être indifféremment en minuscules ou majuscules. Variable=Valeur. L'ordre des variables n'est pas important.

```
Fonte=10
IPv4_PC=127.0.0.1:9999
(...)
```

La modélisation du réseau utilise une nomenclature standard :

Un aiguillage simple est préfixé **A**, suivi de son adresse (exemple : A23)

Un aiguillage triple est noté avec sa première adresse suivi de TRI, suivi de sa deuxième adresse (exemple : 23TRI,25)

Pour les TJD et les TJS, voir plus loin la description.

Un détecteur est simplement noté par son adresse (exemple 517)

Pour les aiguillages, leur position droite est notée **D**, la position déviée est notée **S** et la pointe de l'aiguillage est notée **P**.

La modélisation du réseau est décrite dans le fichier de configuration.

Il y a 5 sections :

La section des variables programme, la section aiguillages, la section branches de réseau, la section feux et la section actionneurs. Chaque section est précédée de son nom entre crochets. **Les variables programmes du fichier sont modifiables directement depuis le programme par le menu « divers/configuration » .**

Nom des sections :

```
[section_aig]
[section_branches]
[section_decodeurs]
[section_sig]
[section_PN]
[section_actions]
[section_dcc++]
[section_trains]
[section_placement]
[section_accCOMUSB]
[section_horloge]
[section_actionneurs]
[section_detecteurs]
```

On y décrit un élément par ligne.

```
(...)
RazSignaux=1
[section_aig]
```

34TRI,27,P516,D514,S515,S2-513,V0,IO
 35TJS,D(530,36D),S(529,36S),L35S,V30,IO
 (...)

Section principale

```

/ Fichier de configuration de signaux_complexes_GL version 8.1
LargeurF=1222                               Largeur de la fenêtre principale
HauteurF=652                                  Hauteur de la fenêtre principale
OffsetX=29                                     Point d'origine X de la fenêtre principale
OffsetY=186                                     Point d'origine Y de la fenêtre principale
Splitter=597                                    Position initiale du splitter
AvecVerifIconesTCO=0                           si 1, vérifie la proximité des icones incompatibles dans le TCO
Algo_localisation=1                            Algorithme de localisation des trains. Seul 1 fonctionne.
Max_Signal_Sens=5                             Nombre maxi d'éléments de rechercher lors d'un signal dans le
                                                bon sens de circulation
Debug=0                                         Débug au démarrage : 0 1 2 ou 3
Mode_Sombre=0                                 Si 1, mode sombre
coul_fond=404040                                Couleur de fond du mode sombre (BVR)
serveurIPCDM_Touche=0                          si 0 : lance le serveur COMIP de CDMrail par la ligne de commande
                                                si 1 : lance par simulation de touches
Filtrage_det=3                                 Filtrage des détecteurs sur retombée, x 100 ms
AntiTimeoutEthLenz=0                           si 1, envoie régulièrement un caractère à l'interface Lenz
                                                connectée en Ethernet
Fonte=10                                       Taille de la fonte de la fenêtre principale
Protocole=1                                     Protocole interface en mode direct :1=XpressNet 2=DCC++
Verif_AdressXpressNet=1                         Vérifie si un accessoire est dans la plage des
                                                adresses interdites en XpressNet
IpV4_PC=127.0.0.1:9999                         Socket CDM rail
ServicesCDM=7                                   Configuration binaire des services demandés à CDM Rail
                                                Bit0=Aiguillages bit1=actionneurs bit2=détecteurs
                                                Bit3=position des trains bit4=signaux
Ipv4_interface=192.168.1.23:5550                  Socket interface centrale
MaxCom=30                                      Valeur maxi du port COM à explorer (valeur maxi = 255)
Protocole_serie=COMX:115200,N,8,1,0             Protocole de l'interface en COM/USB
Inter_car=50                                     Temporisation d'envoi entre deux caractères en COM/USB
Tempo_maxi=15                                    Tempo maxi d'attente avant timeout en COM/USB x 100ms
Entete=1                                         =0 sans entête / =1 entête FF FE
Init_Aig=0                                       si 1 met les aiguillages en position initiale au démarrage
Init_Dem_Aig=0                                 si 1 demande la position des aiguillages à la centrale
                                                temporisation de séquencement entre deux aiguillages
Tempo_Aig=50                                     Demande la connexion à la centrale en COM/USB au démarrage
Init_demUSBCOM=0                                Demande la connexion à la centrale en Ethernet au démarrage
Init_demETH=0                                    =0 fenêtre Moyenne / =1 fenêtre maximalisée
Fenetre=0                                         si 1, positionne la fenêtre principale aux points mémorisés
AffMemoFenetre=0                                Nombre de détecteurs considérés comme trop distants
nb_det_dist=3                                    Vérification de la version au démarrage
verif_version=0                                 Affichage d'information de la version au démarrage
notif_version=0                                Affichage du TCO au démarrage
TCO=1                                            Nombre de TCOs
NbreTCO=1                                       Nom des fichiers des 10 TCOs
Nom_fichier_TCO1=TCO.CFG
Nom_fichier_TCO2=DEUXIEME.CFG
Nom_fichier_TCO3=TCO3.CFG
Nom_fichier_TCO4=TCO4.CFG
Nom_fichier_TCO5=TCO5.CFG
Nom_fichier_TCO6=TCO6.CFG
Nom_fichier_TCO7=TCO7.CFG
Nom_fichier_TCO8=TCO8.CFG
Nom_fichier_TCO9=TCO9.CFG
Nom_fichier_TCO10=TCO10.CFG
MasqueBandeauTCO=0                               si 1 affiche de bandeau de commandes du TCO
CDM=0                                           si 1 lance CDM au démarrage
Lay=RESEAU_GILY_SIGNAL_AJOUTE.LAY               Nom du fichier LAY de CDM
NomModuleCDM=reseau_gily_signal_ajoute_top.cdm   Nom du module du réseau CDM si le réseau a été
                                                Importé de façon automatique

```

```
Serveur_interface=1
retro=1
RazSignaux=0
Tempo_Feu=100
Nb_cantons_Sig=3
Alg_Unisemaf=1
```

Type d'interface connectée à CDM
 Protocole de rétrosignalisation connecté à l'interface CDM
 si 1, envoie un 0 après le pilotage des signaux
 Temporisation de pilotage entre deux commandes aux signaux
 Nombre de cantons présence train avant signal
 Algorithme de pilotage des signaux Unisemaf (1 fonctionne)

Codification des aiguillages simples

Une ligne de la définition d'un aiguillage est constituée des 4 éléments suivants :

Adresse d'aiguillage, **P** (élément connecté à la pointe) , **D** (élément connecté en pos droite) , **S[2-]**, (élément connecté en pos déviée) , Vitesse , Iposition inversée CDM , INIT(position, temporisation, inversion)

« élément » est un détecteur (son adresse) ou un aiguillage (son adresse suivi de S D ou P suivant son point de connexion) – P D et S peuvent être mis dans n'importe quel ordre.

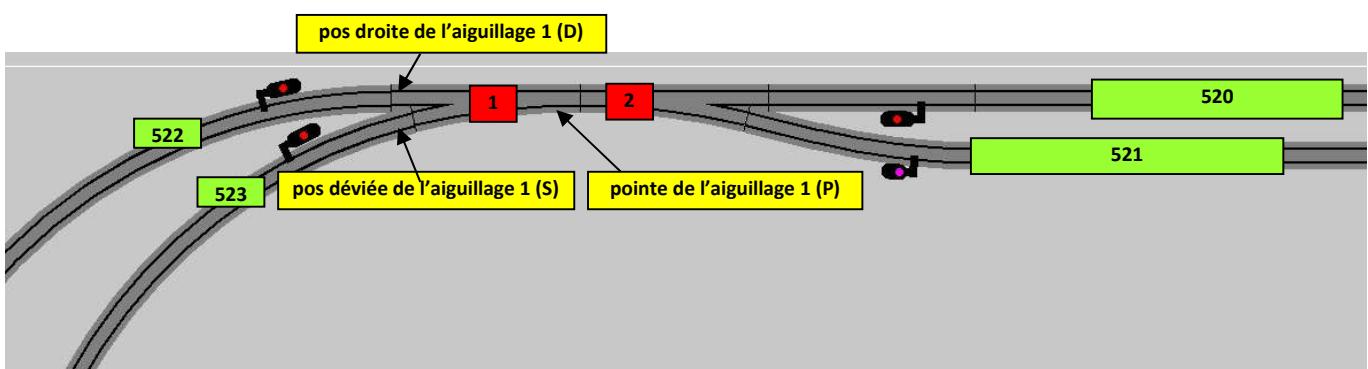
[2-] est dans le cas du branchement à la 2^{ème} position déviée à un aiguillage triple (voir plus loin)

[vitesse] est un élément permettant de décrire la vitesse de franchissement de l'aiguillage en position déviée pour pouvoir afficher le rappel 30 ou 60 sur le signal qui lui est associé. Les valeurs de vitesses autorisées sont V0, V30 ou V60.

[position inversée CDM] : Indiquer I1 si l'aiguillage est piloté en inversé dans CDM (coche "+" devié dans CDM)

Cette information d'inversion est utilisée en mode autonome de signaux_complexes pour inverser l'information déviée et droite de l'aiguillage. Sinon indiquer I0.

INIT décrit le comportement de l'aiguillage en mode autonome au démarrage de signaux complexes (position, temporisation en 1/10 s, inversion de pilotage)



Modélisation des deux aiguillages 1 et 2 ci-dessus :

1, **P2P, D522, S523, V0, I0**
 2, **P1P, D520, S521, V60, I0**

Explication de la ligne 1 :

1 = adresse de l'aiguillage dont la description suit

P2P = signifie : la **pointe** de l'aiguillage **1** est connectée vers la **pointe** de l'aiguillage **2**

D522 = signifie : la position **droite** de l'aiguillage **1** est connectée au détecteur **522**

S523 = signifie : la position **déviée** de l'aiguillage **1** est connectée au détecteur **523**

V0 signifie qu'il n'y a pas de limite de vitesse au franchissement en position déviée.

I0 signifie que la commande de l'aiguillage n'est pas inversée dans CDM rail.

Explications de la ligne 2 :

2 = adresse de l'aiguillage dont la description suit

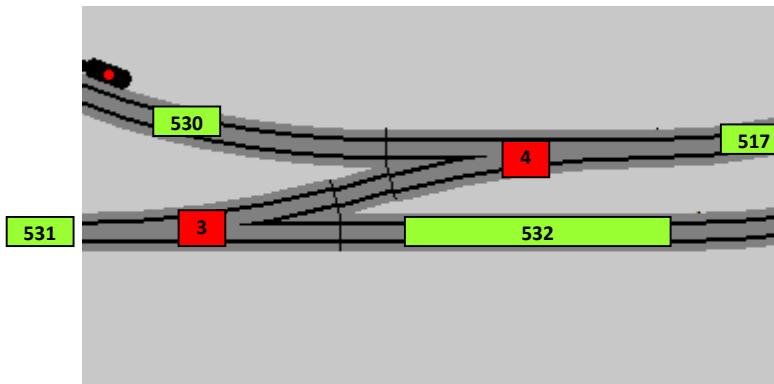
P1P = signifie : la **pointe** de l'aiguillage **2** est connectée vers la **pointe** de l'aiguillage **1**

D520 = signifie : la position **droite** de l'aiguillage **2** est connectée au détecteur **520**

S521 = signifie : la position **déviée** de l'aiguillage **2** est connectée au détecteur **521**

V60 signifie que la vitesse de franchissement en position déviée est de 60 km/h.

Exemple 2 :



<p>Description de l'aiguillage Description de l'aiguillage [3]</p> <p>3,P531,D532,S4S,V0,I0</p> <p>Aiguillage simple <input type="button" value="Restaurer"/></p> <p>Vitesse de franchissement dévié : <input checked="" type="radio"/> sans <input type="radio"/> 30 km/h <input type="radio"/> 60 km/h</p> <p>Représentation</p> <p><input type="checkbox"/> Inversion de l'état CDM</p> <p>Initialisation de l'aiguillage en mode autonome</p> <p>Dévié ou droit <input type="text" value="I"/> Temporisation (1/10ème s) <input type="text" value="5"/></p> <p><input type="checkbox"/> Pilotage inversé</p>	<p>Description de l'aiguillage Description de l'aiguillage [4]</p> <p>4,P517,D530,S3S,V30,I0</p> <p>Aiguillage simple <input type="button" value="Restaurer"/></p> <p>Vitesse de franchissement dévié : <input type="radio"/> sans <input checked="" type="radio"/> 30 km/h <input type="radio"/> 60 km/h</p> <p>Représentation</p> <p><input type="checkbox"/> Inversion de l'état CDM</p> <p>Initialisation de l'aiguillage en mode autonome</p> <p>Dévié ou droit <input type="text" value="I"/> Temporisation (1/10ème s) <input type="text" value="5"/></p> <p><input type="checkbox"/> Pilotage inversé</p>
--	---

Description des aiguillages 3 et 4 à renseigner dans le panneau de configuration

Les aiguillages 3 et 4 auront pour définition :

3, P531, D532, S4S, V0, I0 ...

4, P517, D530, S3S, V0, I0 ...

aiguillage 3 :

P531 = signifie : la pointe de l'aiguillage 3 est connectée au détecteur 531

D532 = signifie : la position droite de l'aiguillage 3 est connectée au détecteur 532

S4S = signifie : la position déviée de l'aiguillage 3 est connectée à l'aiguillage 4 en position déviée

aiguillage 4 :

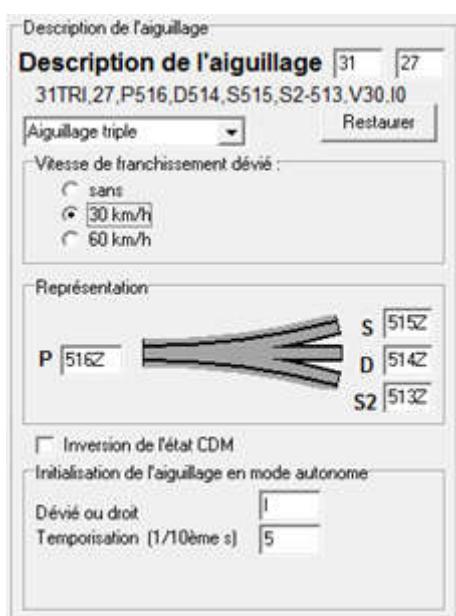
P517 = signifie : la pointe de l'aiguillage 4 est connectée au détecteur 517

D530 = signifie : la position droite de l'aiguillage 4 est connectée au détecteur 530

S3S = signifie : la position déviée de l'aiguillage 4 est connectée à l'aiguillage 3 en position déviée

Attention de ne pas confondre D pour droit avec D comme *dévié* (qui est erroné).

Aiguillage Triple



Description de l'aiguillage triple dans le panneau de configuration.

L'adresse de base est 31, l'adresse secondaire est 27.

Comme chaque aiguillage, on peut définir une vitesse limite de franchissement en position déviée.

Il ne doit bien sûr pas y avoir d'aiguillage 27 préalablement déclaré.

Les éléments D, S, S2 et P peuvent être énumérés dans n'importe quel ordre. Par contre, l'ordre de déclaration de la 1^{ère} adresse et de la 2^{ème} adresse doit respecter le même ordre de déclaration que dans le tableau bleu de CDM-Rail. Ceci est automatiquement réalisé par le remplissage des éléments graphiques/

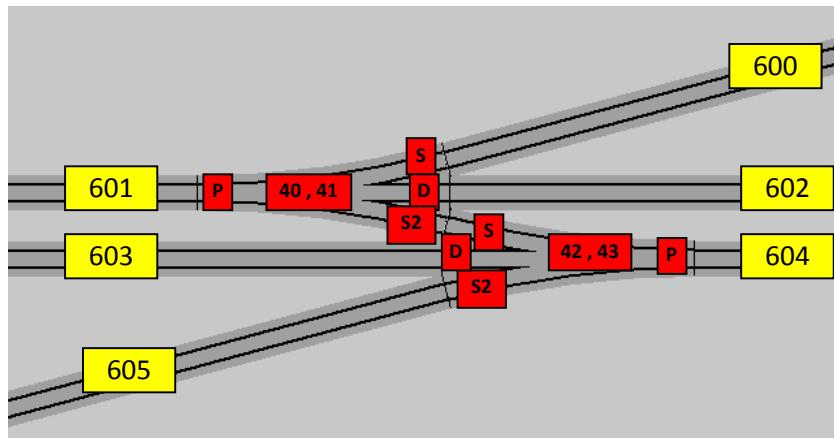
Cet aiguillage sera modélisé de la façon suivante dans le programme des signaux complexes :

31TRI,27,D514,S515,S2-513,P516,V30,I0

Un aiguillage 31 connecté à un la branche S2 d'un aiguillage triple dont la première adresse est 31 sera noté comme suit :

10,P25P,S31S2,D512,V0,I0

Exemple de modélisation de deux d'aiguillages triples



On a ici deux aiguillages triples reliés. Ces deux aiguillages triples sont modélisés comme suit :

40TRI, 41, D602, S600, S2-42S, P601, V0, I0

42TRI, 43, D603, S605, S2-605, P604, V0, I0

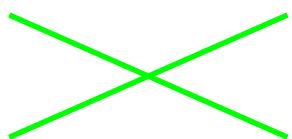
S2-42S signifie que la position déviée 2 de l'aiguillage 40 est connectée à la position déviée 2 de l'aiguillage 42.

Modélisation des TJD

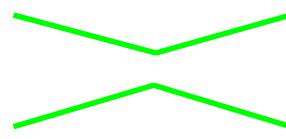
Modélisation des TJD et des TJS

Il y a deux types de TJD: les TJD à 2 états (1 moteur) et les TJD à 4 états (2 moteurs)

TJD2 états :



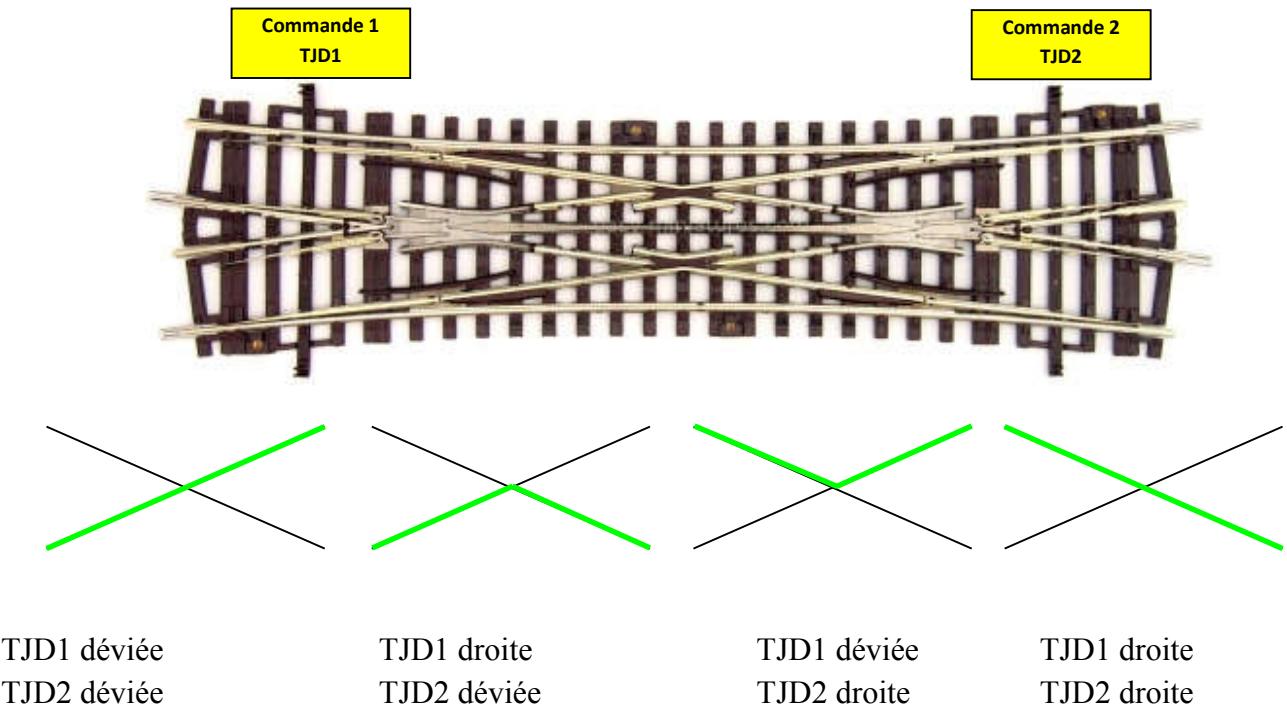
TJD droite



TJD déviée

TJD 4 états :

Elles comportent 2 moteurs pour 4 directions possibles indépendantes :



Une TJD 4 états/TJS dispose de deux adresses. Pour chaque adresse, on renseigne les éléments connectés en position droite (D) et déviée (S). **Une TJD 4 états ou une TJS occupent 2 lignes** dans le fichier de configuration, et donc deux descriptions. Une TJD 2 états n'occupe qu'une ligne.

Modélisation des deux lignes d'une TJD 4 états:

26TJD, D(530, 28D), S(529, 28S), V0, I0, INIT(9, 5), E4

28TJD, D(21D, 26D), S(21S, 26S), V0, I0, INIT(9, 5), E4

Si la TJD est à 2 états, la modélisation est:

26TJD, D(530, 21D), S(529, 21S), V0, I0, INIT(9, 5), E2

notation tjd 4 états:

adresse1_TJD, D(élément ext connecté_D, destination_D), S(élément ext connecté_S, destination_S), Vx, Ix, INIT(x,x), Ex
 adresse2_TJD, D(élément ext connecté_D, destination_D), S(élément ext connecté_S, destination_S), Vx, Ix, INIT(x,x), Ex

élément ext connecté_D = élément extérieur connecté à la TJD1 en position droite

destination_D = destination vers laquelle dirige la TJD1 lorsqu'elle est droite. L'adresse de destination est la 2ème partie de la TJD suivie de D ou S

élément ext connecté_S = élément extérieur connecté à la TJD2 en position déviée

destination_S = destination vers laquelle dirige la TJD1 lorsqu'elle est déviée. L'adresse de destination est la 2ème partie de la TJD suivie de D ou S

Vx : vitesse de franchissement de la TJD (x=0,30 ou 60 km/h)

Ix est un champ qui désigne l'inversion du pilotage de la TJD par CDM Rail. Certaines TJD ont un pilotage inversé dans CDM et doivent donc être inversées par cette notation (x=0 ou 1)

INIT est la description de l'initialisation du positionnement de la TJD (voir section aiguillages simples)

Ex : indique le nombre d'états de la TJD (2 ou 4)

Avec l'inversion:

26TJD, D(530,28D), S(529,28S), V0, I1, INIT(9,5), E4

28TJD, D(21D,26D), S(21S,26S), V0, I1, INIT(9,5), E4

Les TJD qui ne sont pas résolues pendant l'exécution de signaux complexes nécessitent d'être inversées. Il est impossible de prévoir quelles sont celles qui nécessitent une inversion.

ligne 1

26TJD signifie que l'élément à l'adresse **26** est une TJD.

D(530= signifie : l'élément connecté à la position **droite** de la TJD 26 est un détecteur dont l'adresse est 530.

28D) signifie : lorsque la TJD**26** est en position **droite**, elle dirige un convoi vers 28D.

S(529= signifie : l'élément connecté à la position **déviée** de la TJD est un détecteur dont l'adresse est 529.

28S) = signifie : lorsque la TJD26 est en position **déviée**, elle dirige un convoi vers 28S .

ligne 2

28TJD signifie que l'élément à l'adresse 28 est une TJD.

D(21D= signifie : l'élément connecté à la position **droite** de la TJD 28 est l'aiguillage 21 en position droite.

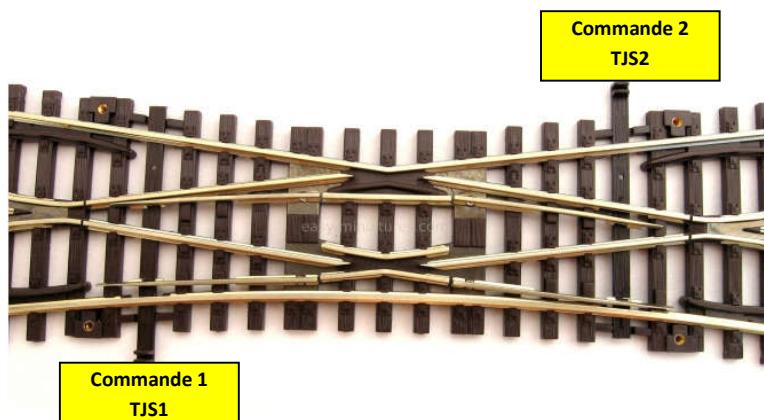
26D) signifie : lorsque la TJD28 est en position **droite**, elle dirige un convoi vers 26D.

S(21S signifie : l'élément connecté à la position **déviée** de la TJD est l'aiguillage 21 en position déviée.

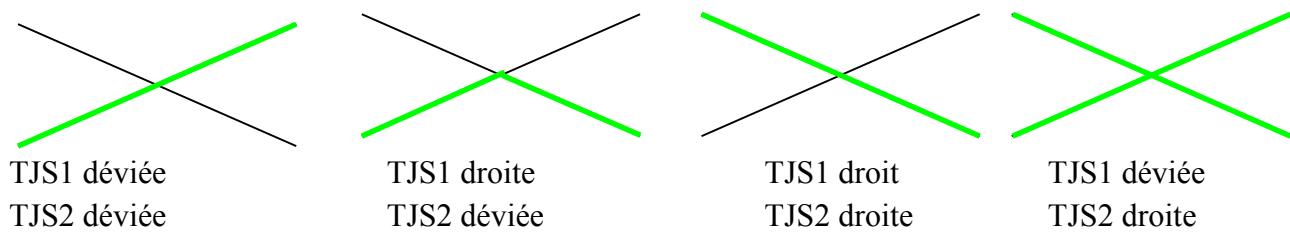
26S) = signifie : lorsque la TJD28 est en position **déviée**, elle dirige un convoi vers 26S .

Pour une TJD à 4 états, les éléments « destination » doivent toujours désigner l'autre partie de la TJD.

Les TJS comportent 2 aiguilles pour 3 directions possibles :

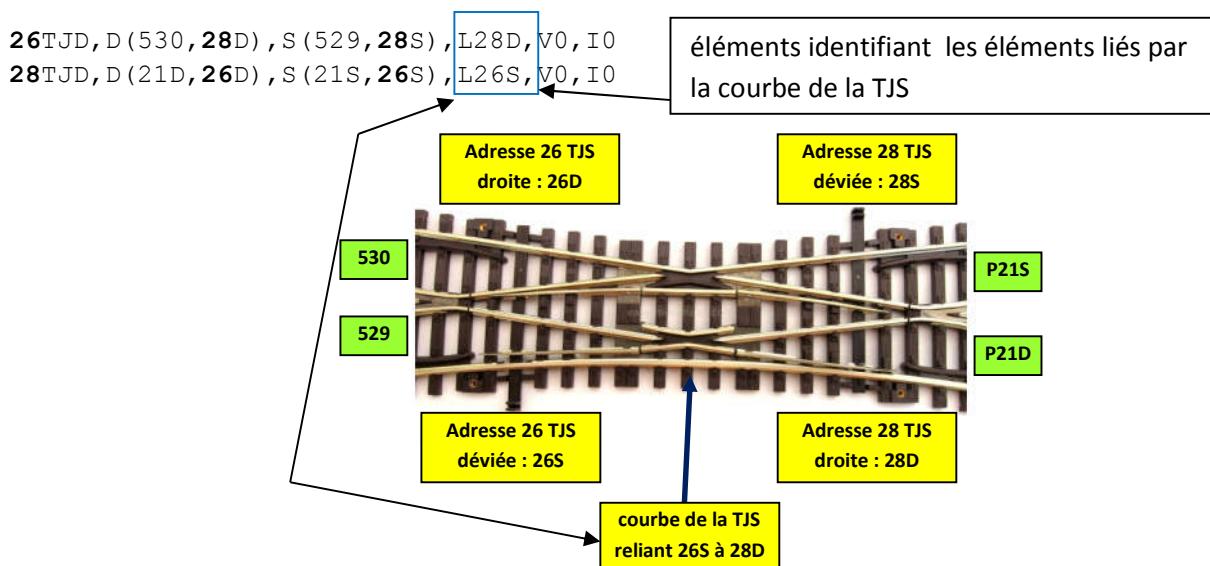


Directions possibles :



Codification d'une TJS

Une TJS comporte un paramètre supplémentaire L dans chaque ligne correspondant à l'endroit liant les éléments de la courbe de la TJS.



Section de modélisation des feux

Les feux sont liés à un détecteur (sauf les feux directionnels). Pour les rendre sensibles au sens de circulation du train, il faut renseigner l'élément suivant immédiatement après le feu (aiguillage ou détecteur). On doit renseigner également la forme du signal (forme de la cible), le type de décodeur qui le pilote et si le feu doit être verrouillable au carré.

Ligne de modélisation :

Il y a une ligne par signal.

Adresse de base du signal , forme, réserve, type de décodeur , (détecteur(s) associé(s) au feu, élément suivant ...), verrouillable , FVCX,FRCX,[Unisemaf],[(conditions en ET pour carré)(conditions en ET pour carré)...],INIT(position,temporisation,inversion de pilotage),SR(...,,)

Adresse du signal

Première adresse sur le bus DCC du décodeur associé au signal

Forme du signal (aspect)

2 = 2 feux (violet blanc) 

3 = 3 feux (rouge jaune vert) 

4 = 4 feux (carré) 

5 = 5 feux (carré + blanc/violet) 

7= 7 feux (blanc/violet + ralentissement) 

9 = 9 feux (blanc/violet + ralentissement +rappel de ralentissement) 

Le signal de rappel sans le ralentissement n'est pas géré, il faut utiliser le signal n°9 à sa place.

Si l'on dispose d'un signal n°7 ou 9 sans carré (sémaphore uniquement) il faut déclarer le signal sans verrouillage au carré.

D2 : indicateur de direction à 2 feux 
Dx jusque x=6 feux

réservé : fonctionnalité réservée au feu blanc.

décodeur

0 = Signal virtuel (correspond à un feu géré mais non implanté et non piloté)

1 = Digitalbahn

2 = CDF

3 = LS_DEC_SNCF

4 = LEB

5 = Digikeiks

6 = Unisemaf de Paco. Ce décodeur nécessite un paramètre supplémentaire dans la ligne, voir le paragraphe spécifique.

7 = Stéphane Ravaut (SR)

8 = Arcmora

9 = LS_DEC_NMBS

10 = B-Models

Les valeurs autres sont les décodeurs personnalisés.

Détecteurs associés, éléments suivants

liste de détecteurs et des éléments suivants (max 4) associés au signal séparés par une virgule. Voir plus loin : signal associé à plusieurs voies ; le tout entre parenthèses

FVCX, FRCX : indique si le signal doit afficher un feu vert clignotant (X=1) ou/et un feu rouge clignotant (X=1)

[Unisemaf]

Variable supplémentaire décrivant la cible dans le cas d'un décodeur unisemaf uniquement. Doit être précédée d'un U.

[(conditions en ET pour carré)() ..]

Champs optionnels permettant au signal d'afficher un carré en fonction de la position de certains aiguillages. Chaque élément entre parenthèses désigne un ou plusieurs aiguillages avec leur position (S ou D) en condition ET. Les parenthèses suivantes enchaînent des conditions en OU. Voir exemple plus loin.

[SR]

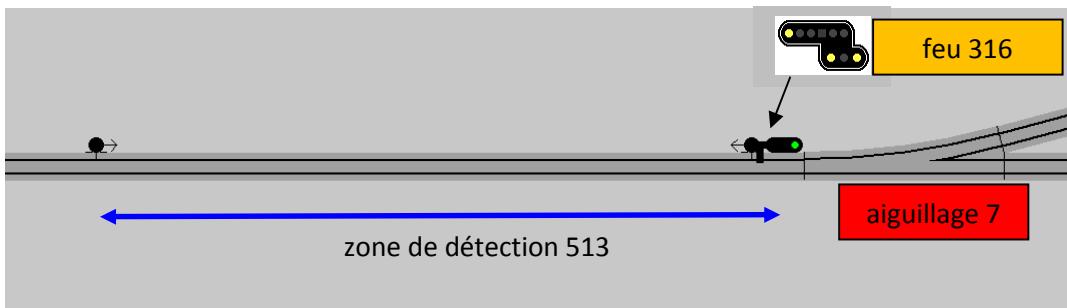
Champ décrivant le paramétrage des 8 adresses (commande1 commande0) donc 16 paramètres du décodeur Stéphane Ravaut.

161, 4, 0, 7, (538, A32), 0, SR(adr1-1, adr1-0, adr2-1, adr2-0, adr3-1, adr3-0, adr4-1, adr4-0, adr5-1, adr5-0, adr6-1, adr6-0, adr7-1, adr7-0, adr8-1, adr8-0)

[MOT]

Champ décrivant les sorties envoyées aux décodeurs digikeijs et CDF.

Exemple de codification pour le signal ci – dessous :



Le feu d'adresse 316 à 9 feux est piloté par un décodeur digitalBahn (1) est associé à la zone de détection 513. L'élément immédiatement suivant après le feu est l'aiguillage 7, et le feu est verrouillable au carré. Il sera modélisé comme suit :

316, 9, 0, 1, (513, A7), 1

A7 est l'aiguillage 7 rencontré immédiatement après le feu dans le sens de circulation de visibilité du feu. S'il n'y a pas d'aiguillage mais un détecteur, on note simplement l'adresse du détecteur.

Exemple d'un feu avec conditions supplémentaires sur le carré

161, 4, 0, 4, (538, A32), 0, (A21S, A6D) (A30S, A20D) (A1D, A2S, A3D)

Cette définition du feu 161 contient des éléments permettant au feu d'afficher un carré si les aiguillages désignés ont la position indiquée, c'est-à-dire :

(aiguillage 21 dévié et aiguillage 6 droit) ou (aiguillage 30 dévié et aiguillage 20 droit) ou (aiguillage 1 droit et aiguillage 2 dévié et aiguillage 3 droit)

En mode RUN avec CDM, une temporisation de 2s est traitée si un train est arrêté en face d'un feu rouge lors du redémarrage du train.

Section Actionneurs

Cette section décrit les actions devant être déclenchés lorsqu'un train active ou désactive un actionneur pendant le RUN de CDM rail. Il existe deux types d'actions possibles :

Les actions pour *Fonctions F* et les actions pour les *passages à niveaux* (PN).

Cette section comporte une ligne par action, sa notation dépend de l'action.

Actionner une fonction F d'une locomotive (F1 à F12)

Le déclenchement de l'action « Fonction F » est provoqué sur le nom de train déclaré dans les descriptions de convois de CDM.

Adresse_Actionneur , état (0 ou 1) , nom du train , Action , temporisation de retombée en ms .

Adresse actionneur :

Adresse de l'actionneur ou du détecteur, ou Zone de détection : MEM[527,520]

Action :

Fx = numéro de fonction.

Nom du train :

Nom du train défini dans CDM rail ou X pour que la condition s'applique à tous les trains.

Exemple 1 :

815,1,CC406526,F4,400

Déclenche la fonction F4 lorsque l'actionneur 815 est mis à 1 par le train CC406526. La fonction F4 retombe après 400 ms.

Ce qui se traduit par l'envoi de la commande F4=1 puis F4=0 400 ms plus tard.

La fonction F4 à 1 envoie un coup de klaxon.

Exemple 2 :

815,1,X,F4,400

Comme ci-dessus, mais l'action sera déclenchée pour chaque train.

Exemple 3:

Mem[527,520],1,X,"Klaxon_2.wav"

Joue le son "Klaxon_2.wav" quand un train entre dans la zone des détecteurs de 527 à 520. Le champ X n'est pas interprété.

Actionner un passage à niveau à une ou plusieurs voies

Le déclenchement de l'action « PN » est provoqué par n'importe quel train.

(Voie 1 Adresse actionneur provoquant la fermeture, Voie 1 Adresse actionneur provoquant l'ouverture) ,

(Voie 2 Adresse actionneur provoquant la fermeture, Voie 2 Adresse actionneur provoquant l'ouverture) , (Voie n,)...,

PN(adresse de fermeture du PN+-,adresse d'ouverture du PN+-)

Exemple passage de deux voies sur un PN :

(815,830), (820,840), PN(121+,121-)

Voie 1 Voie 2

Actionner un accessoire depuis un actionneur

Permet de piloter un accessoire depuis un actionneur.

Adresse_Actionneur , état (0 ou 1) , nom du train , **A**Adresse_Accessoire , valeur de la sortie (1 ou 2) , commande de remise à 0 ou de maintien (S ou Z)

Nom du train :

Nom du train défini dans CDM rail ou X pour que la condition s'applique à tous les trains.

Exemple 1 :

815,1,CC406526,A613,1,S

Lors du passage du train CC406526 sur l'actionneur 815 à 1 (activation), on envoie 1 à l'adresse d'accessoire 613 et la sortie reste à 1 (S).

Exemple 2 :

815,1,X,A613,2,Z

Lors du passage de n'importe quel train sur l'actionneur 815 à 1 (activation), on envoie 2 à l'adresse d'accessoire 613 puis la sortie passe à 0 (Z).

Attention : Si l'accessoire est un aiguillage sans remise à 0, ne pas piloter un aiguillage à bobine, il sera alimenté en permanence ce qui entraînera sa destruction.

Ne pas oublier le **A** dans l'adresse d'accessoire, sinon la commande ne sera pas traitée.

Section décodeurs personnalisés

```
[section_decodeurs]
/ décodeur n°1
Nom_dec_pers=Personnalise1      Nom du décodeur personnalisé
NombreAdresses=4                 Nombre d'adresses occupées par le décodeur
Nation=1                          Nationalité 1=France 2=Belgique
Commande=0                         0= commandé par la centrale - 1= Commandé par un périphérique COMUSB socket
Periph=1                           Numéro de commande du périphérique
```

```

1,2,0,1,2          aspect 1, aspect 2, décalage d'adresse, sortie DCC1 pour aspect1,
3,4,1,1,2          sortie DCC2 pour aspect2,
5,9,2,1,2
7,8,3,1,2
/ décodeur n°2
Nom_dec_pers=decodeur USB Belge
NombreAdresses=10
Nation=2
Commande=1          Commandé par un périphérique COMUSB socket
Periph=2
vert jaune horizontal,vert jaune H aspect, commande de cet aspect
rouge,rouge
vert,vert
vert jaune vertical,
rouge blanc,
deux jaunes,
Chiffre,
Chevron,
Clignote,
0

```

Section DCC++

```

[section_dcc++]
EnvAigDccpp=0      Si 1, envoie la liste des aiguillages à la centrale DCC++
AdrBaseDetDccpp=513 Adresse de base à ajouter aux détecteurs renvoyés par la centrale
[init_dcc++]
<Y 8 1>
<1>
0

```

Section Trains

Contient la liste des trains de l'onglet "Trains"

```

[section_trains]
BB25531,1,120,50,40      Nom du train, adresse du décodeur, vitesse maximale, nominale, réduite.
TGV,2,120,50,30
BB16024,3,120,60,50
CC406526,4,120,80,60
CAMERA,0,120,0,0
train,99,120,60,40
0

```

Section Trains

Contient la liste de placement de trains de la fenêtre placement des trains.

```

[section_placement]
BB25531,0,0,0      Nom du train, détecteur recevant la loco, détecteur destinataire, sens
TGV,0,0,0
BB16024,0,0,0
CC406526,523,526,0
CAMERA,0,0,0
TRAIN,0,0,0
0

```

Section Périphériques

Contient la liste des périphériques COM/USB et socket.

```

[section_accCOMUSB]
accessoire,0,0,0,1,1,COM10:115200,n,8,1      Nom périph, service aiguillages, service détecteurs,
periph_usb,0,0,0,0,1,com13:9600,n,8,1          services actionneurs, mode visible, envoi CR, protocole.
Socket,0,0,0,0,0,192.1.2.250:1500

```

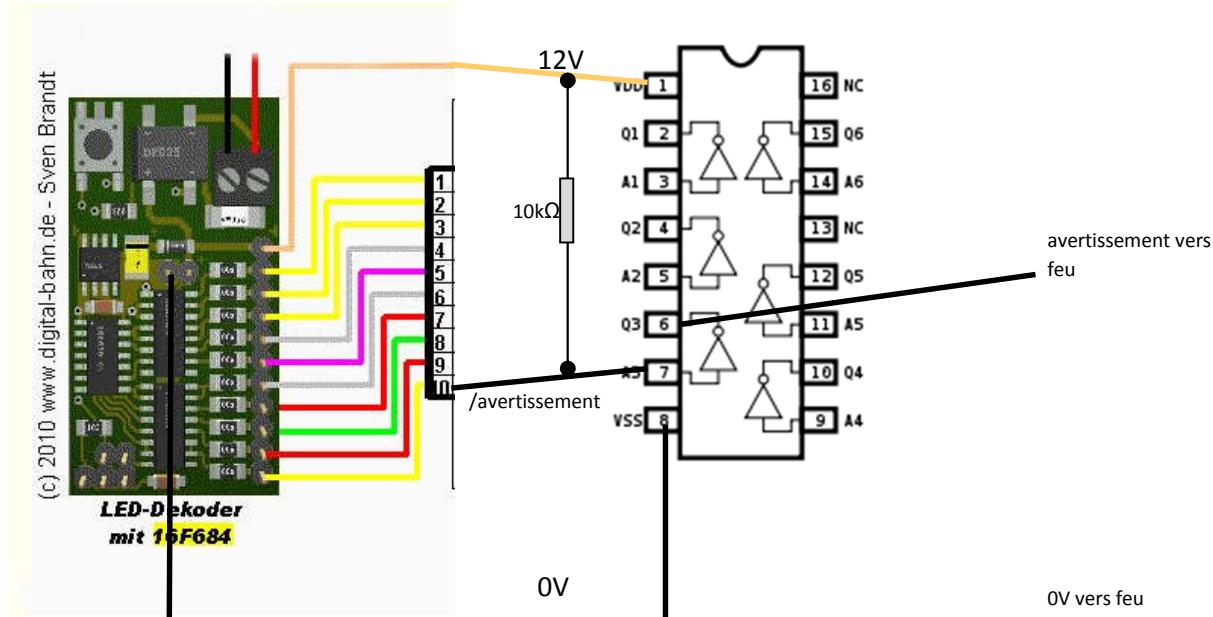
Arrêt des logiciels

Pour l'arrêt les logiciels, procéder dans l'ordre :

1. Arrêter le mode Run,
2. Arrêter le serveur Comm IP sur CDM Rail, ou fermer CDM rail
3. Fermer le programme client.

Annexe 1 : interface permettant l'utilisation de signaux avec commun au +

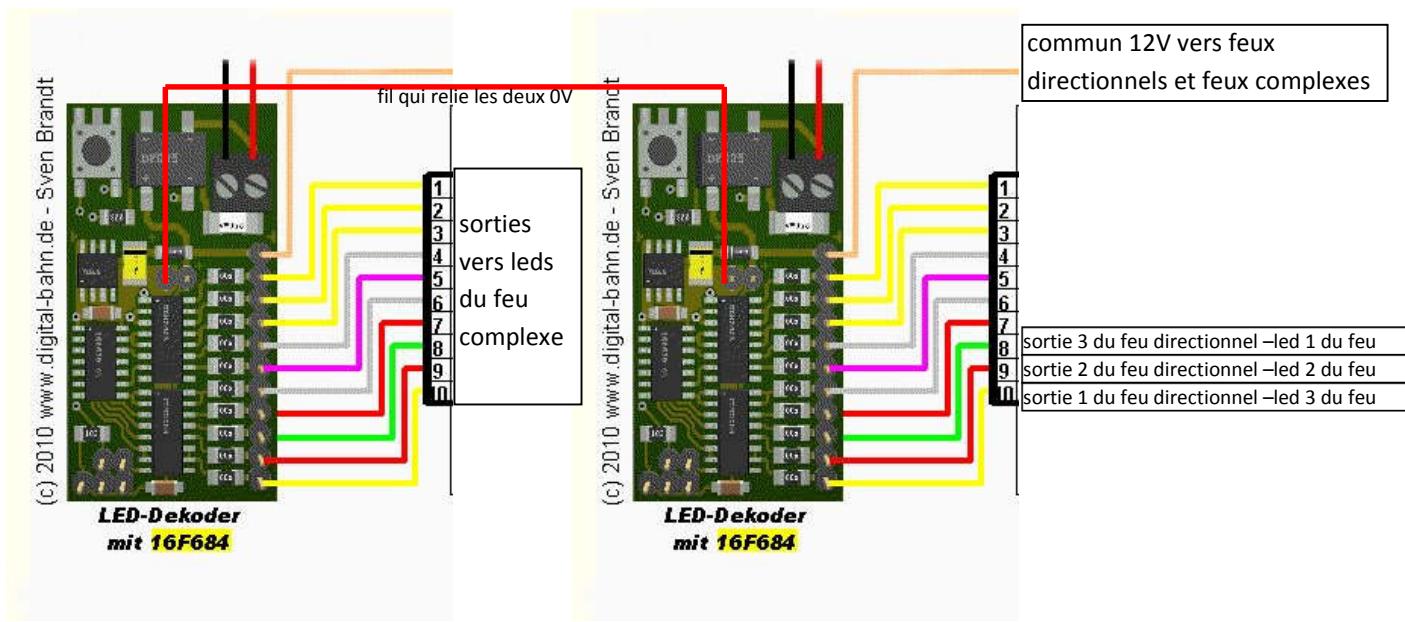
Ci dessous figure le schéma d'une interface permettant l'utilisation de signaux avec un commun au + avec les décodeur digitalbahn avec un circuit CMOS 4049 (6 inverseurs)



L'exemple ne montre le câblage que pour l'avertissement (1 état). Le 4049 inverse l'état du signal vers le feu. Il faut câbler chaque signal souhaité. La résistance de 10kΩ est une résistance de tirage au VCC car les sorties de l'ULN2003 du décodeur sont des sorties à collecteurs ouverts. Le 4049 permet le câblage de 6 états. Pour câbler un feu de plus de 6 états, il faudra utiliser deux 4049.

Annexe 2 : câblage des décodeurs pour un feu complexe et feux directionnels (décodeurs digitalbahn)

Pour piloter un feu muni d'un panneau de feux directionnels avec un seul commun au +, il faut utiliser deux décodeurs dont il faut relier les 0V: (point milieu du bornier X5 à 3 points) :



décodeur du feu complexe
équipé du logiciel d'exploitation des feux complexes

décodeur de direction
équipé du logiciel led shalten

Annexe 3 : guide de choix des signaux et des décodeurs

1. Choisir les signaux avec un commun au +, sachant qu'ils seront utilisés avec un décodeur. Seul le décodeur LDT peut piloter à la fois des signaux avec commun au + ou au -.
2. Certains signaux ne différencient pas le carré du sémaphore (les 2 leds rouges de ces signaux sont câblées en parallèle sur un fil, donc ils s'allument forcément en même temps). Il faut donc bien se renseigner sur le fait que la led du carré est dissociée de la led sémaphore, cela dépend du fabricant. Certains fabricants artisans acceptent la modification à la commande.

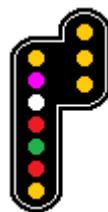
3. Cette documentation propose une description sur l'intégration des décodeurs DigitalBahn , CDF ,LEB et LDT. Chacun possède ses avantages et inconvénients listés dans le tableau ci-dessous.

caractéristique	marque	avantage	inconvénient
prix et boîtier	CDF	décodeur monté, 30€	
	DigitalBahn		décodeur en kit CMS, 13€
	LDT	en kit 43€ ou monté 57€	
	LEB	19€	
protocole	CDF		motorola non
	DigitalBahn	DCC & motorola (suivant logiciel embarqué dans le pic)	
	LDT	DCC et Motorola	
	LEB	DCC	motorola non
connexion des signaux	CDF	bornier à vis	
	DigitalBahn		Bornier HE10. kit pour bornier à vis
	LDT	bornier à vis	
	LEB	connecteur pour signaux LEB	ne fonctionne qu'avec des signaux LEB
configuration	CDF	par variables de configuration CV	
	DigitalBahn	par reprogrammation du PIC ou par bouton	la reprogrammation par pic nécessite un programmeur de pic. Les signaux de programmation sont disponibles.
	LDT	par bouton	
	LEB	par variables de configuration CV, il faut utiliser un bouchon de programmation	
utilisation	CDF		adressage variable en fonction du mode et du câblage
	DigitalBahn	adressage fixe	
	LDT		adressage variable en fonction du mode et du câblage
	LEB	adressage fixe	
étendue d'adressage d'un décodeur	CDF	4 adresses ; de 1 à 4 signaux par décodeur	
	DigitalBahn	14 adresses ; un seul signal par décodeur	
	LDT	8 adresses ; de 2 à 8 signaux par décodeur	
	LEB	8 adresses	
alimentation des signaux	CDF	12V	
	DigitalBahn	12V	
	LDT		5V + R330 ohms (pour 12V, kit module adap LSA)
	LEB	12V	
commun d'alimentation des signaux	CDF	signaux avec commun au + exclusivement	
	DigitalBahn	signaux avec commun au + exclusivement	
	LDT	signaux avec commun au + ou au -	
	LEB	signaux LEB avec commun au - exclusivement	
représentation de tous les états d'un signal complexe	CDF		représentation maximale : 8 feux à 6 états
	DigitalBahn	oui (possible sur un signal 10 feux à 19 états)	
	LDT		la présentation du rappel 30 ou 60 combinée avec l'avertissement ou l'avertissement clignotant n'est pas possible.
	LEB	oui (possible sur un signal 10 feux à 19 états)	
compatibilité avec toutes les centrales DCC	CDF		non, sauf si on utilise le décodeur exclusivement en mode 0
	DigitalBahn	oui	
	LDT	oui	
	LEB	oui	
gestion de l'œilletton	CDF	oui avec matrice à diodes externe	

	DigitalBahn	oui	
	LDT		non
	LEB	oui	
gestion des panneaux directionnels	CDF	oui, en mode 2	
	DigitalBahn	oui avec logiciel led_shalten dans le pic du décodeur	
	LDT		non
	LEB	oui	

Annexe 4 - Liste des états d'un signal SNCF complet à 10 feux

signal hypothétique complet à 10 feux :



Etats possibles de ce signal :

- | | |
|--|---|
| 1. Carré | arrêt absolu |
| 2. Sémaaphore | arrêt ou franchissable |
| 3. Sémaaphore clignotant | franchissable à 15km/h, peut annoncer un carré |
| 4. Vert | voie libre |
| 5. Vert clignotant | limitation à 160km/h |
| 6. Carré violet | arrêt absolu sur voie de garage ou de service |
| 7. Blanc | départ en manœuvre autorisé |
| 8. Blanc clignotant | départ en manœuvre limitée autorisé |
| 9. Avertissement | annonce un carré, un sémaaphore, ou un sémaaphore clignotant |
| 10. Avertissement clignotant | annonce un avertissement ou un ralentissement implantés à faible distance |
| 11. Ralentissement 30 | annonce un rappel 30 |
| 12. Ralentissement 60 | annonce un rappel 60 |
| 13. Ralentissement 60 + avertissement clignotant | annonce un rappel 60 et un avertissement |
| 14. Rappel 30 | franchissement d'aiguille à 30 km/h |
| 15. Rappel 60 | franchissement d'aiguille à 60 km/h |
| 16. Rappel 30 + avertissement | franchissement d'aiguille à 30 km/h et annonce un signal rouge ou un nouveau rappel de ralentissement |
| 17. Rappel 30 + avertissement clignotant | franchissement d'aiguille à 30 km/h et annonce un avertissement |
| 18. Rappel 60 + avertissement | franchissement d'aiguille à 60 km/h et annonce un signal rouge ou un nouveau rappel de ralentissement |
| 19. rappel 60 + avertissement clignotant | franchissement d'aiguille à 60 km/h et annonce un avertissement |