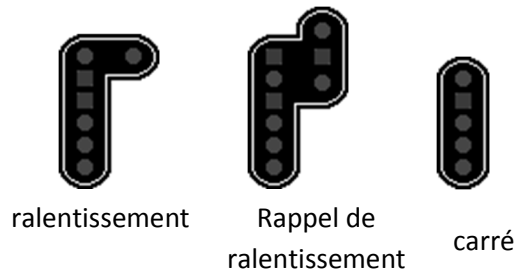


Principe d'utilisation du kit client IP de CDM rail pour utiliser les décodeurs de signaux digitalbahn , CDF, LEB ou LDT (ou d'autres)

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 V10 » qui pourra être utilisé comme point de départ, téléchargeable [ici](#)
- CDM rail V5.1 mini.
- décodeur de feux « led dekodeur » 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** (1)
- L'environnement de développement Delphi7
- éventuellement, un programmeur de PIC pour reprogrammer les adresses du décodeur de digital bahn.

(1) pour utiliser d'autres décodeurs, voir page 32, paragraphe Utilisation d'autres décodeurs

Introduction	4
Exemple d'implantation	5
Exemples de situations	6
Règles de présentation des signaux combinés	7
Règles de présentation des feux directionnels	8
Exemple d'implantation	8
Principe de pilotage	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
Feux directionnels / TIV	17
Décodeurs Digital Bahn	17
Décodeurs CDF	18
Restrictions et spécificités du programme client pour les signaux	19
Configuration du décodeur Digitalbahn pour les signaux complexes	20
Configuration par programmeur	20
Configuration « manuelle » du décodeur de feu digitalBahn (si on ne dispose pas de programmeur)	22
Décodeur DigitalBahn : Configuration du décodeur d'adresses, sans utiliser toutes les adresses	23
Fichiers PIC pour le décodeur DigitalBahn	23
Codes commandes pour le décodeur digitalBahn	24
Logiciels PIC pouvant équiper le décodeur digital Bahn	24
Mise en place du décodeur DigitalBahn	25
Utilisation du décodeur CDF80108007S pour les signaux complexes	26
Utilisation du décodeur LDT-DEC-SNCF pour les signaux complexes	28
Utilisation du décodeur LEB pour les signaux complexes	30
Utilisation d'autres décodeurs	32
Configuration du décodeur Digitalbahn pour les feux directionnels ou les TIV	33
Description logicielle du programme client SIGNAUX_COMPLEXES	34
Modification du programme avec Delphi 7	34

Installation des composants socket (TClientSocket et TServerSocket)	34
Installation du composant MScomm32	34
Debugueur	34
Défaillances possibles	35
Généralités sur la détermination des routes sur les évènements détecteurs	38
Gestion de la centrale LENZ	39
Gestion des routes	40
Liste des structures de données utilisées	43
<i>Procédures et fonctions</i>	45
<i>Insertion de code pour un signal spécifique</i>	46
<i>Fonctionnement de l'échange de données entre le client et le serveur</i>	47
<i>Exemple de programmation du kit client</i>	48
<i>Utilisation du programme client</i>	48
<i>Pilotage d'essai</i>	48
<i>Annexe 1 : interface permettant l'utilisation de signaux avec commun au -</i>	50
<i>Annexe 2 : câblage des décodeurs pour un feu complexe et feux directionnels (décodeurs digitalbahn)</i>	51
<i>Annexe 3 : guide de choix des signaux et des décodeurs</i>	52
<i>Annexe 4 - Liste des états d'un signal SNCF complet à 10 feux</i>	55

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.

Le programme client signaux complexes calcule les états des signaux complexes en fonction des informations de détection du réseau transmis depuis CDM rail.

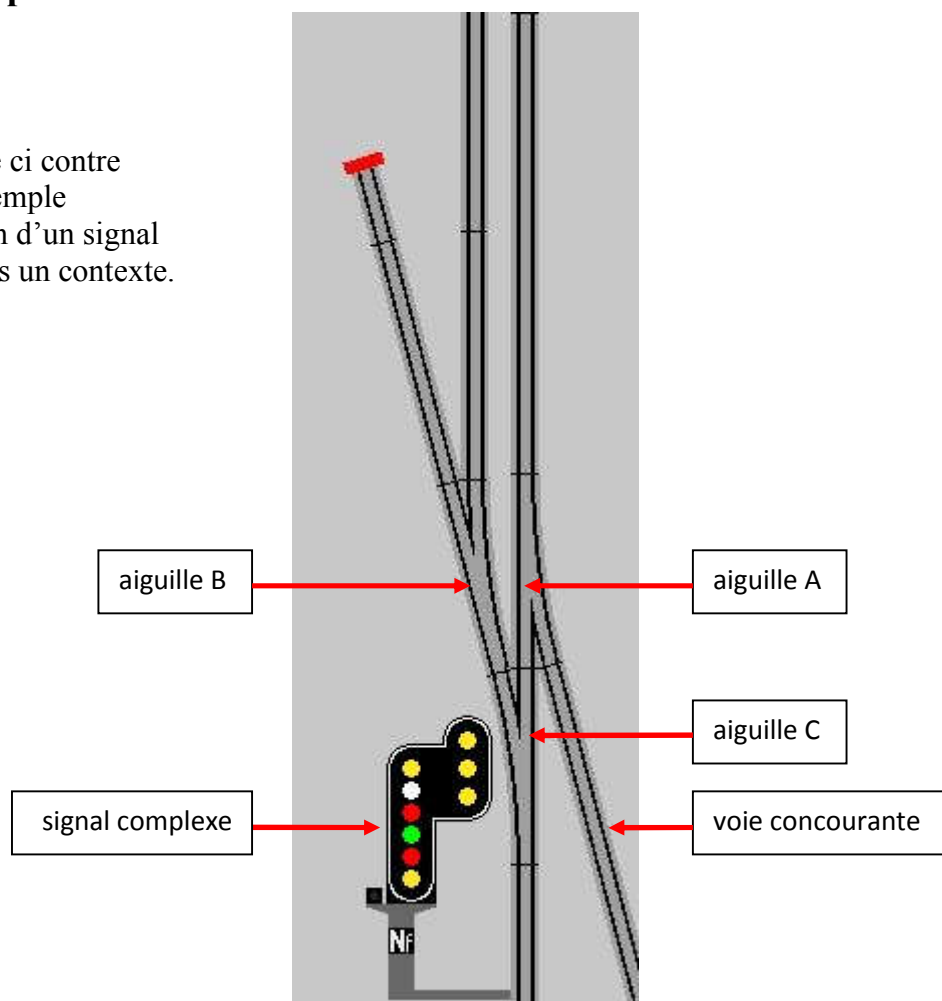
Il met en mémoire les détecteurs actionnés, puis tente de les associer pour vérifier que deux détecteurs actionnés l'un après l'autre l'ont été par le même train ou non.

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

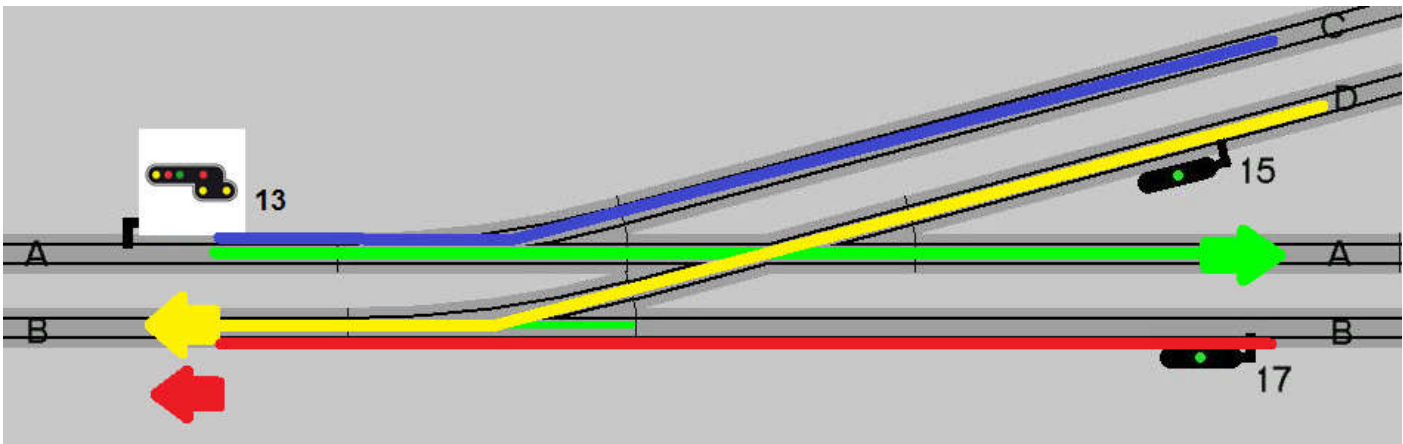
Le synoptique ci contre montre un exemple d'implantation d'un signal complexe dans un contexte.



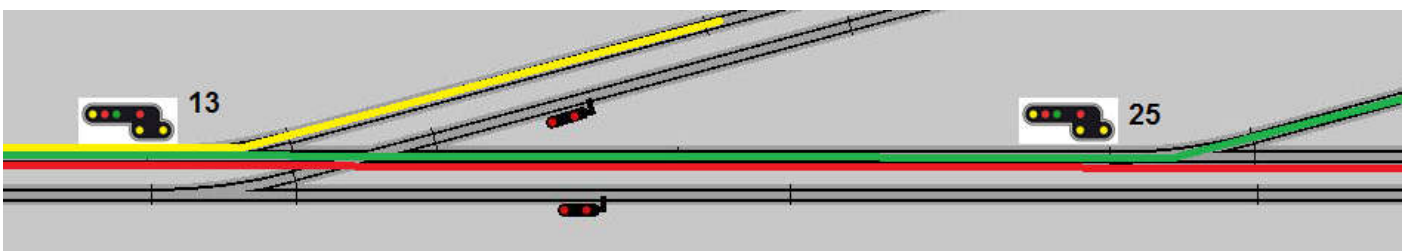
présentation du signal	règles de présentation du signal
carré	-aiguille A pas en direct -signal de la voie concourante ne présentant pas de carré (voie ouverte) -pas de présence train sur les cantons précédant le signal
sémaphore	présence train sur les cantons précédent le signal et train présent sur 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 CDM, il peut être provoqué par un élément du menu du programme client.
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 feu 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 feux 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 préavertissement 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 préavertissement 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 . Dans ce cas il faut gérer la coordination de ces 2 signaux dans le code.

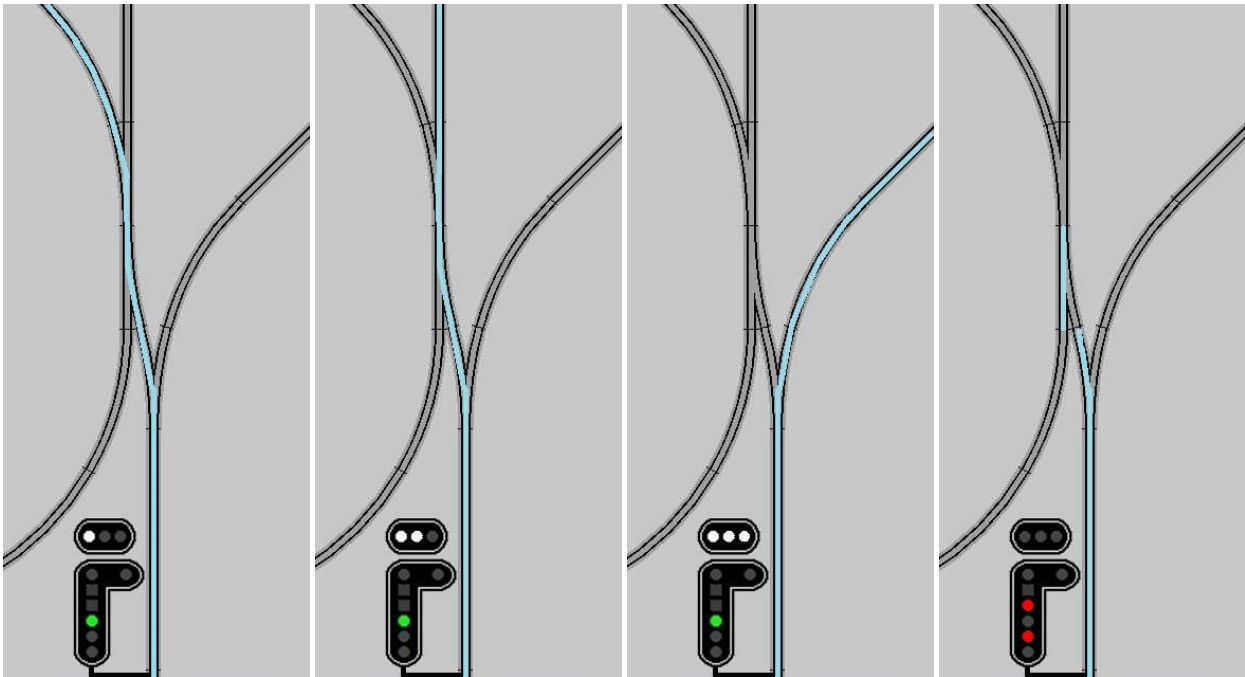
Décodeurs Digital Bahn : tous les états sont possibles, mais il faut gérer l'extinction de l'avertissement sur un ralentissement 30 (géré dans l'organigramme du programme client)

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 feux 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

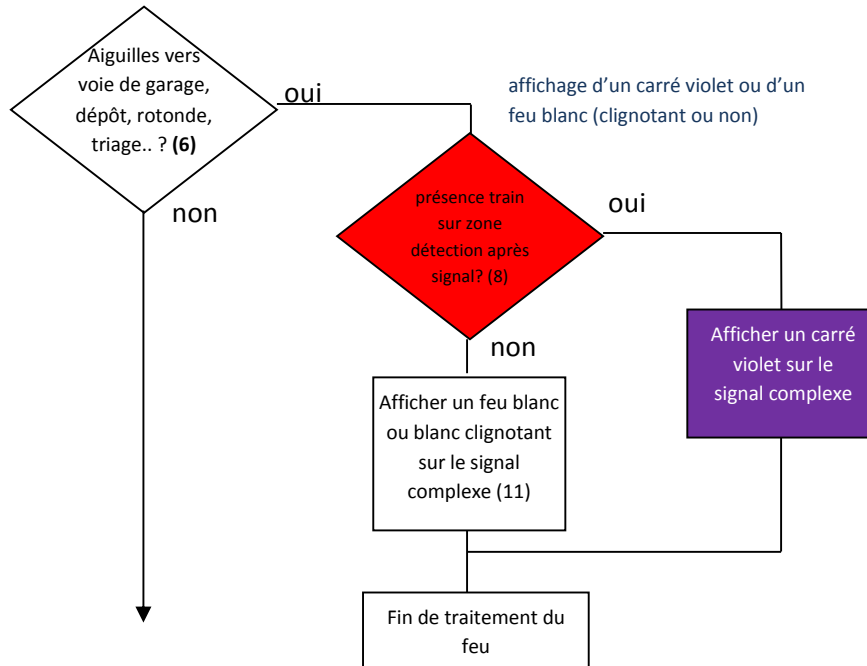
Direction invalide car la position de l'aiguillage prise en talon n'est pas positionnée dans le sens du déplacement : le feu présente donc un carré et l'indicateur de direction est éteint.

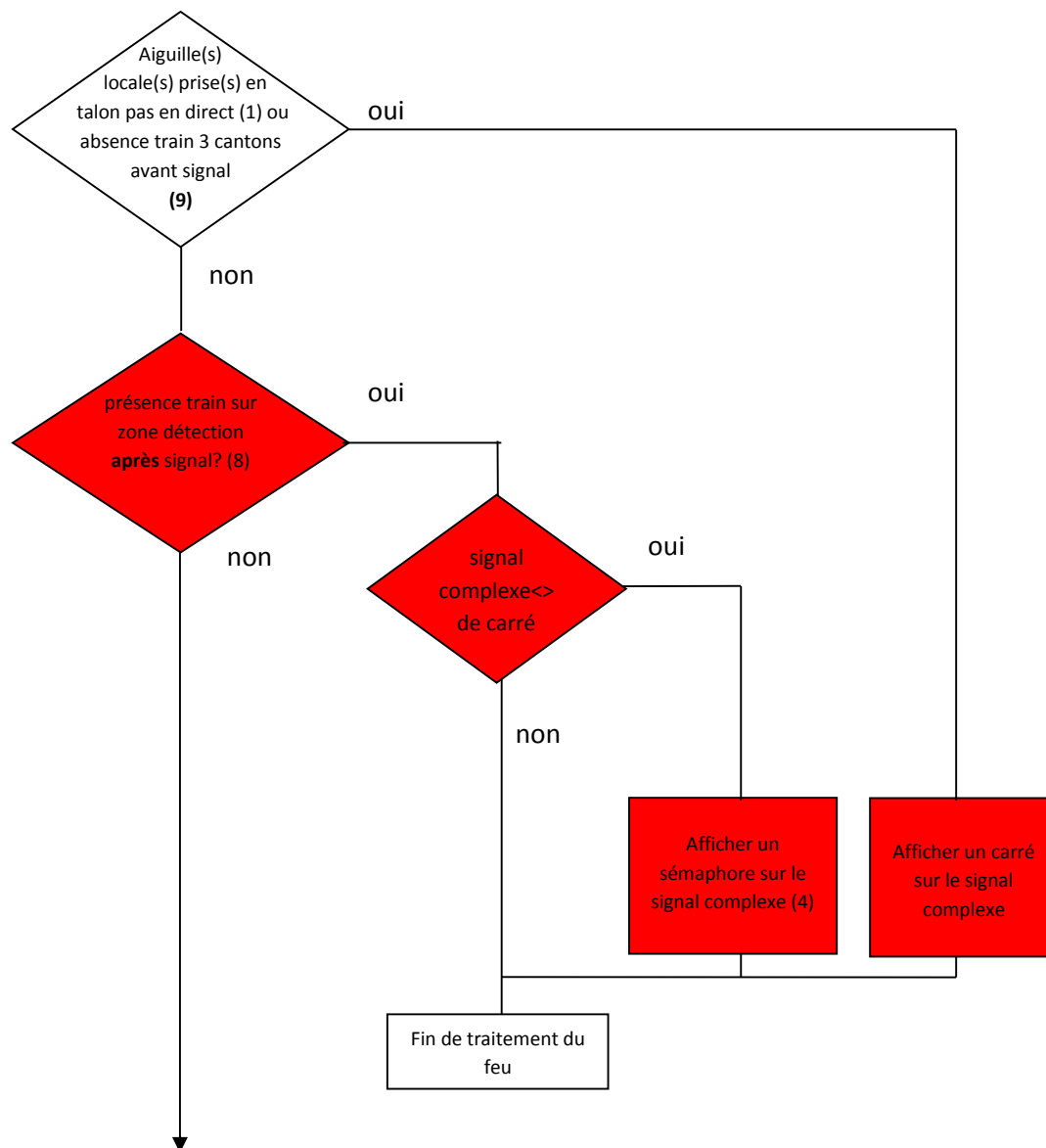
Le pilotage des sorties se fait en fonction des aiguillages qui dirige le train vers la position géographique désignée par le feu directionnel. Le nombre de feux directionnels dépend du nombre de positions géographiques de destination.

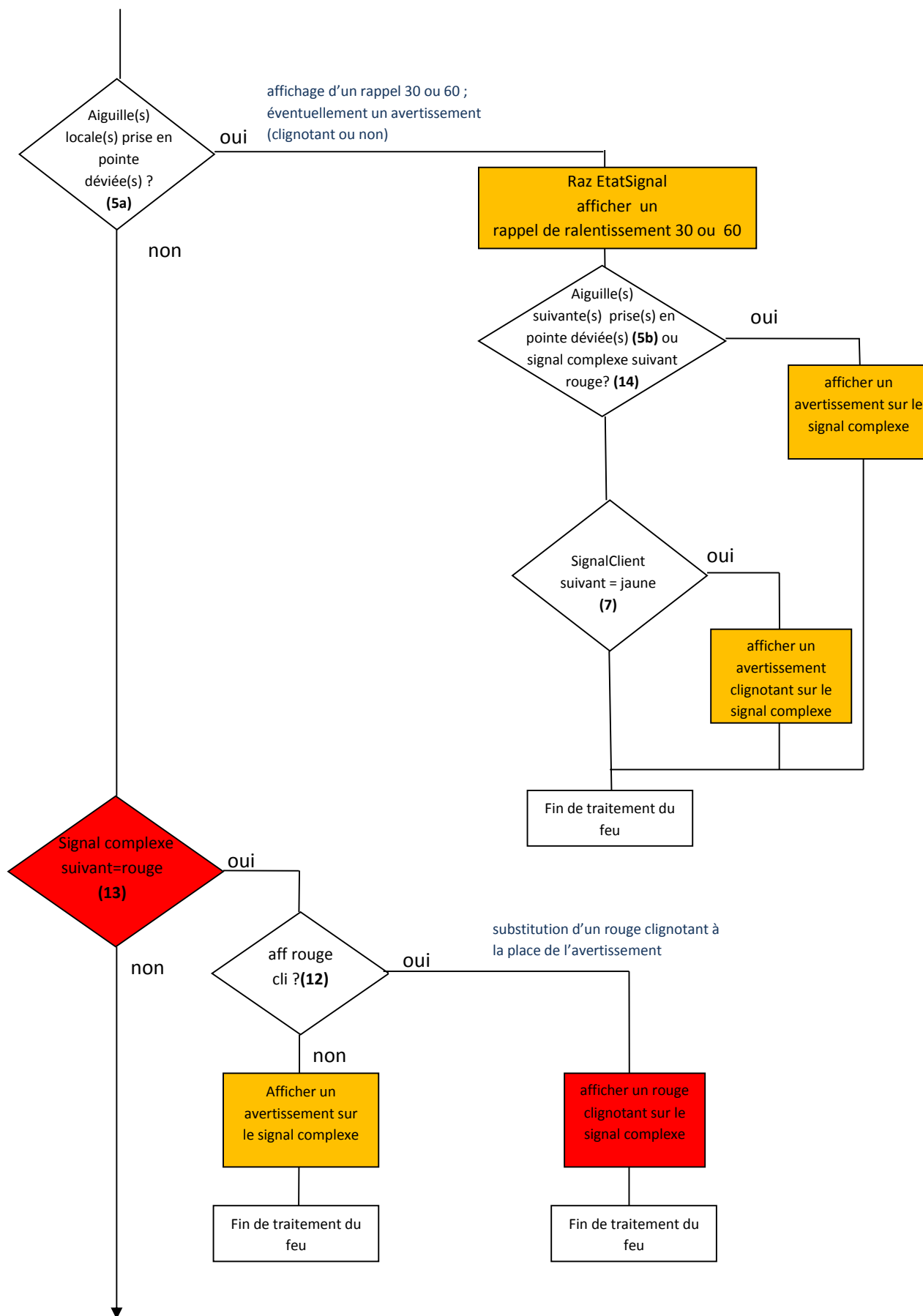
NB. Le feu complexe présenté ci dessus pour exemple n'a pas la forme permettant de représenter un rappel de ralentissement car les aiguillages pris en pointe permettent dans ce cas de les franchir à plus de 60 km/h. Un TIV conditionnel doit donc être implanté avant les aiguillages.

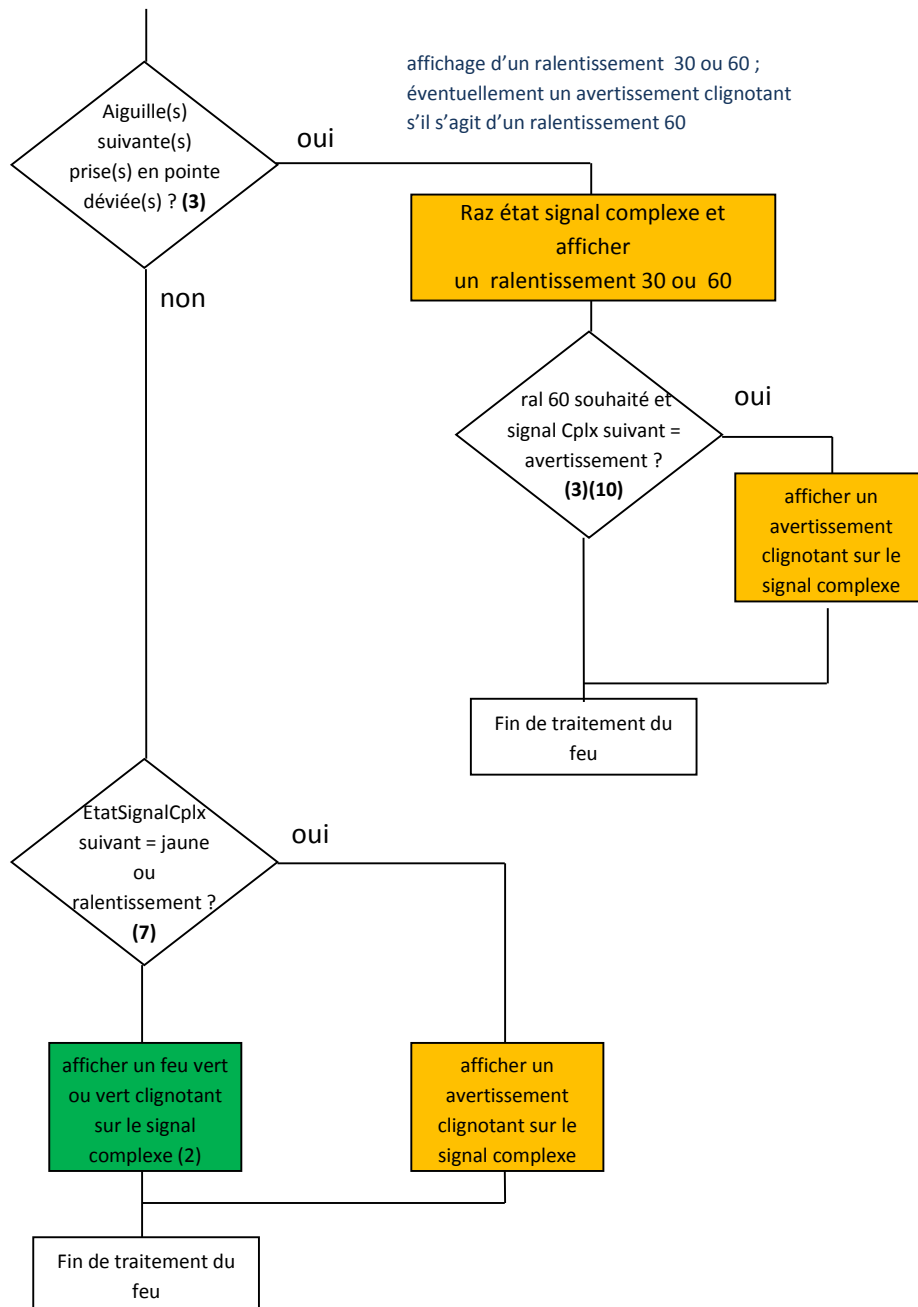
Principe de pilotage

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 par exemple être affiché si le train qui se présente dans les cantons précédant le signal est un TGV et que le feu est un feu de pleine voie. Pour interroger le nom du train devant un actionneur, il faut utiliser la ligne suivante : `if (strcmp(actionneur[813], "TGV")==0) ...etc..` à condition que le nom du train soit « TGV » dans CDM.
- (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)
- (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 30 ou 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 substituer l'avertissement (feu jaune) de CDM avec le sémaphore clignotant.
- (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.

PARAMETRES ACTIONNEUR

Côté de l'actionneur

Direction

Ident.

VISIBLE ☒

CANCEL OK

Paramètres de l'actionneur :
 edition module/édition des
 signaux du module
 le n# Ident n'est pas obligatoire

**CONFIG. ACTIONNEUR
ADRESSE DIGITALE**

ADRESSE

ACCESS. ACTION

ACC. STD. ☒

OUT1 ☐ OUT2 ☐

LOCO ACTION

LOCO ☐

ON ☒ OFF ☐

PULSE ☐

Déconfigurer

Aide Annul. OK

configuration de l'actionneur :
 configuration/configuration
 d'actionneur.

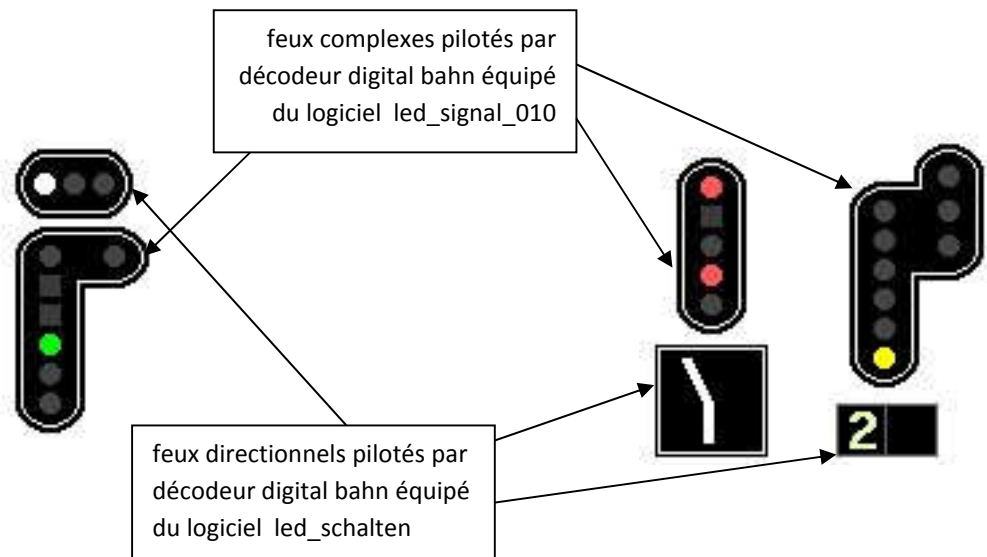
renseigner l'adresse qui sera
 utilisée dans le programme client.
 De base, un actionneur sert à
 piloter un accessoire comme ici, la
 coche de ACC STD est présente et
 envoie une commande sur la sortie
 OUT 1 de l'adresse 800.

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 :

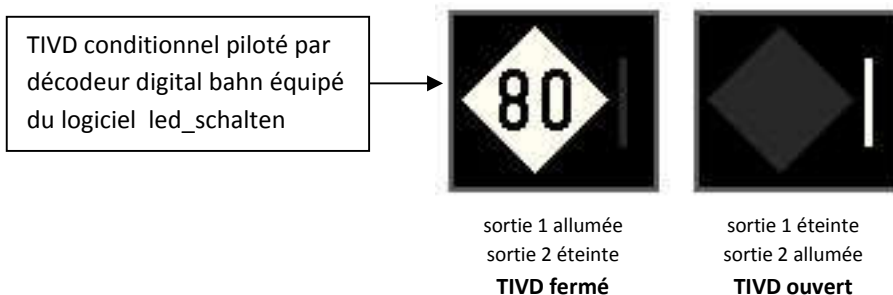


Exemple :

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 :



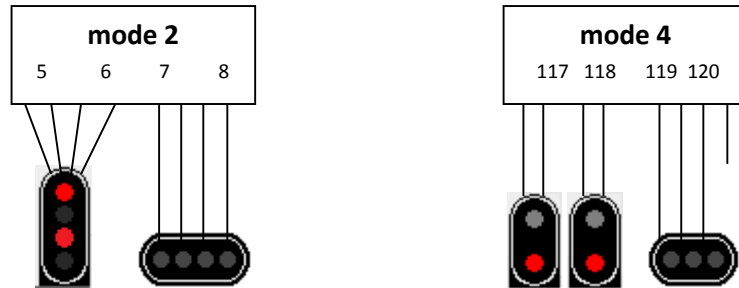
NB : TIVD=tableau indicateur de vitesse à distance, en annonce d'un aiguillage distant pris en compte dont la vitesse de franchissement est différente de 30 ou 60 km/h pour lequel on aurait implanté un signal complexe.

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 :

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

Envoi_DirectionCDF(119,3) ; 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.

Décodeurs DigitalBahn : en principe, on ne peut pas implanter plus de 146 signaux complexes. En effet la liste des adresses permises dans le décodeur de Digital-Bahn va de 1 à 2044. Néanmoins, sur les décodeurs dont on n'utilise pas toutes les adresses, il est possible de programmer l'adresse poubelle « 0 » et de repartir à l'adresse suivante pour le décodeur suivant. Ceci permet de sauvegarder des adresses et d'implanter plus de 146 décodeurs, ou de restreindre le plan des adresses. Voir note page 20.

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 événements 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 programmeur 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 programmeur ou en modifiant directement manuellement les adresses.

Configuration par programmeur

(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 programmeur 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

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) ou 0 (rouge). Laisser vert par défaut pour compatibilité avec le programme client.

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 :

`IPC_AccessoryDCCParamCRE (NULL, adresse, 2, IPC_COMMAND) ;`

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.

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.

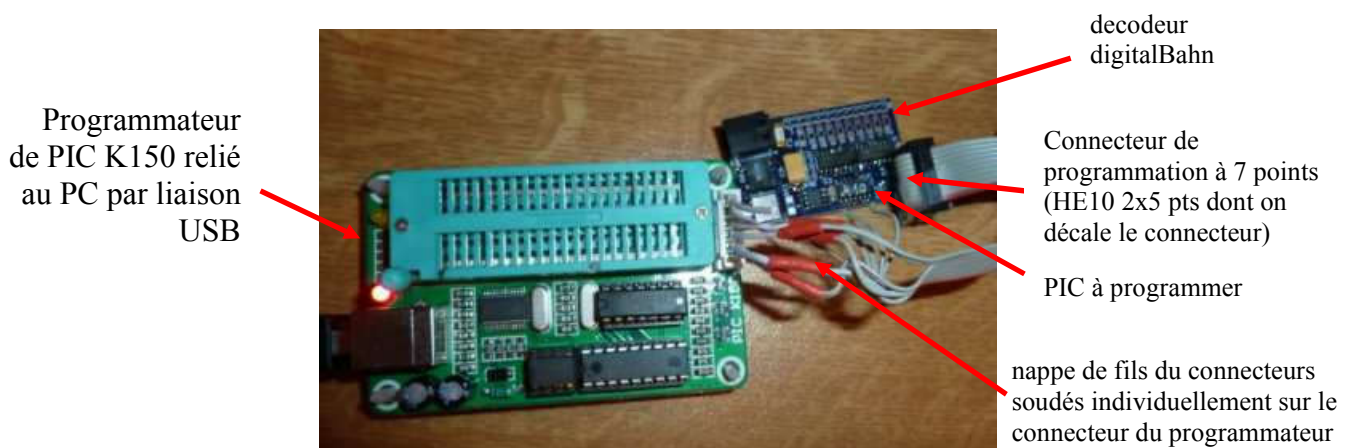
NB3 : Pour afficher un ralentissement 30 sur un signal présentant déjà un avertissement, il faut envoyer un sémaphore pour effacer le ralentissement 30 ;

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

Configuration du décodeur de feu DigitalBahn par programmeur (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 programmeur de PIC. Par exemple le **K150**. (photo ci dessous). On réinjecte le fichier HEX dans le programmeur. Le pic est programmé sur la platine du kit après qu'il soit soudé. Ensuite dans le logiciel de programmation (par exemple microBrn), 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.



Configuration « manuelle » du décodeur de feu 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

Décodeur DigitalBahn : Configuration du décodeur d'adresses, sans utiliser toutes les adresses

Cette configuration permet de préserver les adresses non utilisées sur un signal complexe. L'exemple ci-dessus montre la configuration d'un signal complexe doté de 3 feux + ralentissement. On ne désire pas non plus utiliser le feu vert clignotant, ni le carré violet ni le feu blanc. Ces fonctions sont donc remplies avec l'adresse poubelle « 0 ». La dernière adresse de ce signal étant 83, le signal suivant commencera donc en 84. On peut laisser quelques adresses libres si on souhaite utiliser des fonctions supplémentaires plus tard ; mais le signal peut tout à fait recevoir des adresses non contigües.

Hex Manipu V0.991

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.	0	R/G
Cv - violet	0	R/G
M - white	0	R/G
(M) - white flash.	0	R/G
A - yellow	0	R/G
(A) - yellow flash.	0	R/G
30 R - slow	82	R/G
60 (R) - slow	83	R/G
30 RR - remember	0	R/G
60 (RR) - remember	0	R/G

0 est l'adresse poubelle à affecter aux fonctions non utilisées du décodeur affecté au feu complexe.
Ici, les fonctions validées sur ce signal sont :
Carré, sémaphore,
sémaphore clignotant, vert ,
ralentissement 60 et
ralentissement 30

Grâce à cette astuce, on n'utilise que 6 adresses au lieu des 14 de base pour un signal qui ne dispose que de 6 fonctions ☺. **Néanmoins, cette solution ne permet pas de standardiser le pilotage dans le programme client par la fonction : `envoi_signalbahn(int adresse,int code)`.** Chaque feu aura dans ce cas des fonctionnalités à des adresses différentes ce qui complique l'écriture du programme client et ne permet pas d'utiliser un mode de fonctionnement standardisé et paramétré. Il faudra donc écrire une fonction de pilotage par feu.

Fichiers PIC pour le décodeur DigitalBahn

Les sources des PIC sont disponibles ici :

http://www.digital-bahn.de/develop/source/dekoder_asm_files.zip

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 :

<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

PIC LED-Dekoder "LED_Signal_010" ne pas oublier de choisir le protocole 3,00€
pic684-410 DCC ou Motorola
http://www.digital-bahn.de/shop/product_info.php?products_id=91

Bausatz LED-Dekoder, ohne PIC platine avec composants, sans pic 9,80€
http://www.digital-bahn.de/shop/product_info.php?products_id=106

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

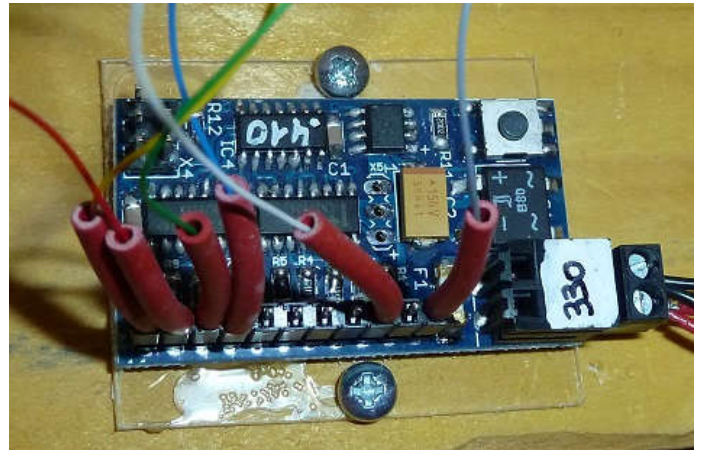
Il existe d'autres logiciels fournis par DigitalBahn sur leur site pour d'autres effets d'allumage qui peuvent être utilisés pour l'ambiance de l'éclairage d'un réseau (exemples : simulation allumage de néons, simulation de poste à souder, clignotement pour les feux de passage à niveau...). Tous ces 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 aisé 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 (tri et tétrachloroéthylène) 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

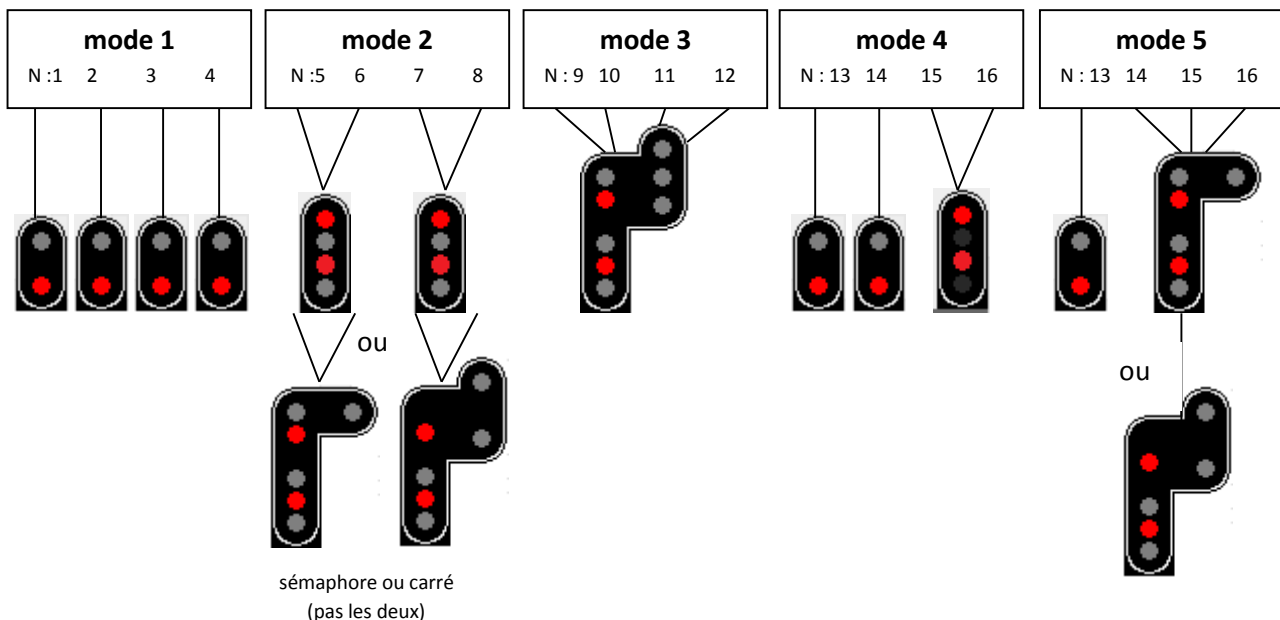
Mode 2 : commande de 2 signaux à 4 états

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

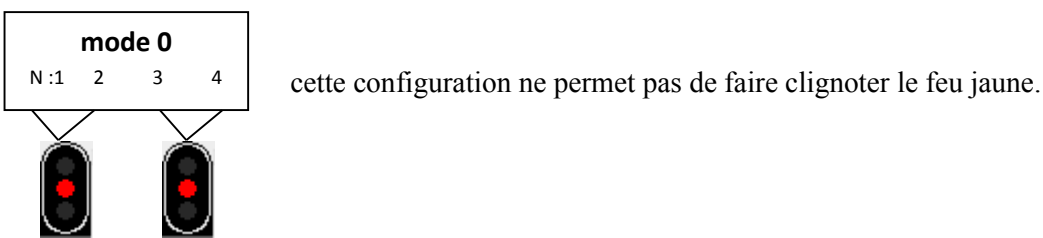
Mode 4 : commande de 2 signaux à 2 états, un signal 4 états

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 :



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**.

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. Devant la diversité de câblage pour un mode et plusieurs feux, **chaque utilisateur devra écrire le pilotage de chaque signal dans la procédure Envoi_cdf en fonction du câblage du feu :**

Procédure pour piloter les signaux branchés sur les décodeurs CDF :

Envoi_CDF(adresse)

adresse : adresse du signal sur le bus

exemple : la procédure suivante permet de piloter un signal « à 4 feux » sur le décodeur CDF à l'adresse de base 476 :

Extrait de la procédure envoi_CDF() :

```
(...)
aspect=code_to_aspect(code)
switch (adresse)
case 476 : // pilotage du signal à l'adresse 476
{
    if (aspect==carre)      IPC_AccessoryDCCParamCRE(NULL_adresse,2,IPC_COMMAND);
    if (aspect==semaphore) IPC_AccessoryDCCParamCRE(NULL_adresse,1,IPC_COMMAND);
    if (aspect==vert)      IPC_AccessoryDCCParamCRE(NULL_adresse+1,2,IPC_COMMAND);
    if (aspect==jaune)     IPC_AccessoryDCCParamCRE(NULL_adresse+1,1,IPC_COMMAND);
}
(...)
break;
```

2 correspond à la touche – de la LH100 chez Lenz

1 correspond à la touche + de la LH100 chez Lenz

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)

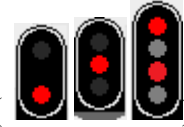
aspect=code_to_aspect(code)

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.

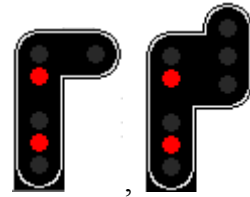


2 adresses permettent de piloter un signal à 2 feux, 3 feux ou à 4 feux. (

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

commandes
du
groupe 1

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

puis

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

commandes
du
groupe 2

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	commandes du groupe 3
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	commandes du groupe 4
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	

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)

Utilisation d'autres décodeurs

Pour utiliser un autre décodeur, il faut copier la procédure « `envoi_signalbahn(adresse)` » (pour les signaux complexes) et la procédure « `envoi_direction` » (pour les tableaux indicateurs de direction) dans le module `principal.c` du programme client. Ces deux procédures doivent être écrites en fonction des spécificités de programmation du décodeur souhaité :

```
procedure envoi_signalclient(adresse : integer) ;  
...  
    Pilote_acc(adresse,code,feu) ; ←ligne(s) à modifier en fonction du décodeur
```

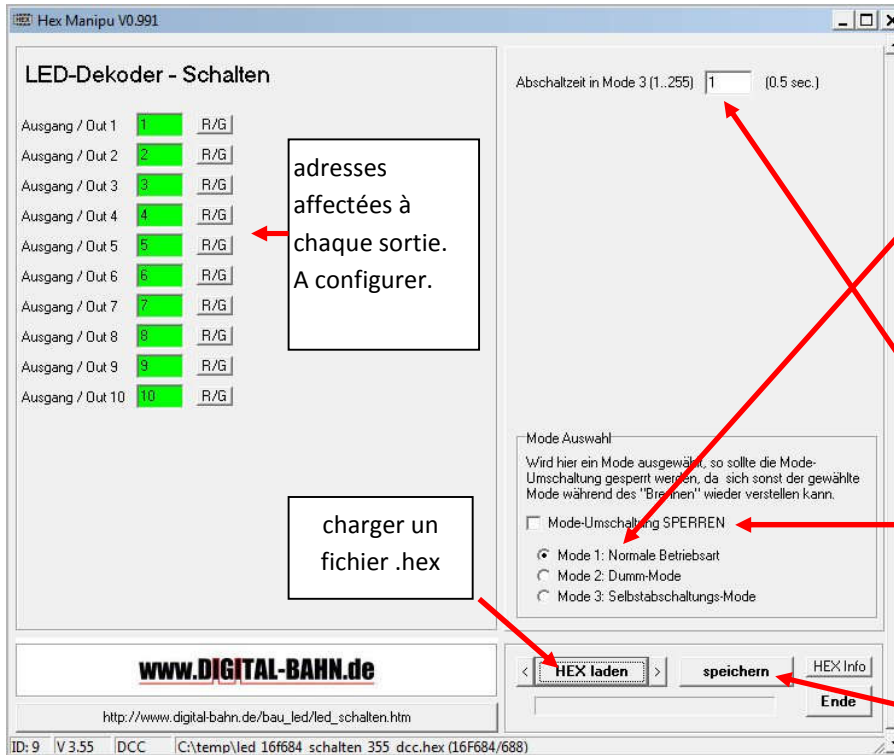
adresse est l'adresse de base du décodeur de signal.

code est la valeur à envoyer au décodeur pour réaliser l'allumage désiré sur le signal.

Feu est un type énuméré qui indique à la procédure `Pilote_acc` que l'on pilote un feu et non un aiguillage.

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 programmeur de PIC.



Configuration au démarrage du décodeur :

Mode 1 : fonction marche / arrêt des leds, le décodeur démarre avec son état de la mise hors tension précédente.

Mode 2 : le décodeur démarre toujours avec ses sorties à 0.

Mode 3 : le décodeur fonctionne en mode impulsion. Chaque sortie est activée pendant un temps X configurable par pas de 0,5s.

Ce champ est accessible ici

Verrouillage de mode. Si coché, le changement de mode avec le bouton au démarrage du décodeur n'est pas possible.

sauver le fichier hex modifié, qui sera utilisé pour reprogrammer le pic du décodeur

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 ;

IPC_AccessoryDCCParamCRE (NULL, **adresse**, 2, IPC_COMMAND) ; allumage led 1

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

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

Pour une configuration manuelle sans programmeur, voir « Configuration « manuelle » du décodeur de feu », 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

Modification du programme avec Delphi 7

Pour les programmeurs, signaux complexes GL a été écrit en Delphi7. Cette plateforme 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

Il reste à configurer Delphi pour y intégrer les composants additionnels Sockets et TmsComm.

Installation des composants socket (TClientSocket et TServerSocket)

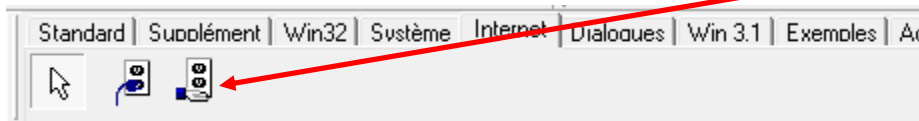
Menu composants / installer les paquets, cliquer sur ajouter

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 ServerSocket apparaissent dans l'onglet Internet

Installation du composant MScomm32

Menu composants / importer un contrôle activeX / cliquer sur ajouter

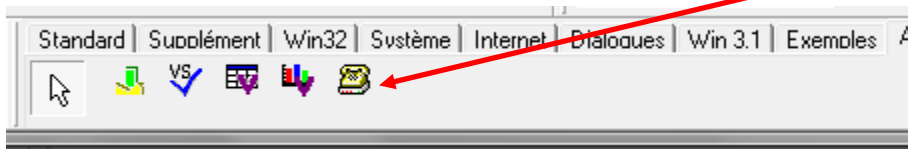
chercher le fichier mscomm32.ocx (qui est dans le répertoire signaux_complexes_gl)

cliquer sur ok

Dans la liste, sélectionner « Microsoft Comm Control 6.0 (Version 1.1) »

Cliquer sur installer puis sur cliquer unité

Sauvegarder



Le composant TMSComm apparaît dans l'onglet ActiveX

Debugueur

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

Défaillances possibles

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

Fermer Delphi

Editer en ascii le fichier UnitPrinc.dfm

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

Réouvrir Delphi 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: TMSComm;)

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 = {
    2143341208000000ED030000ED03000001568A640000060000000010000040000
    00020000802500000000080000000000000000000003F00000011000000}
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 :

```
regssvr32 mscom32.ocx
```

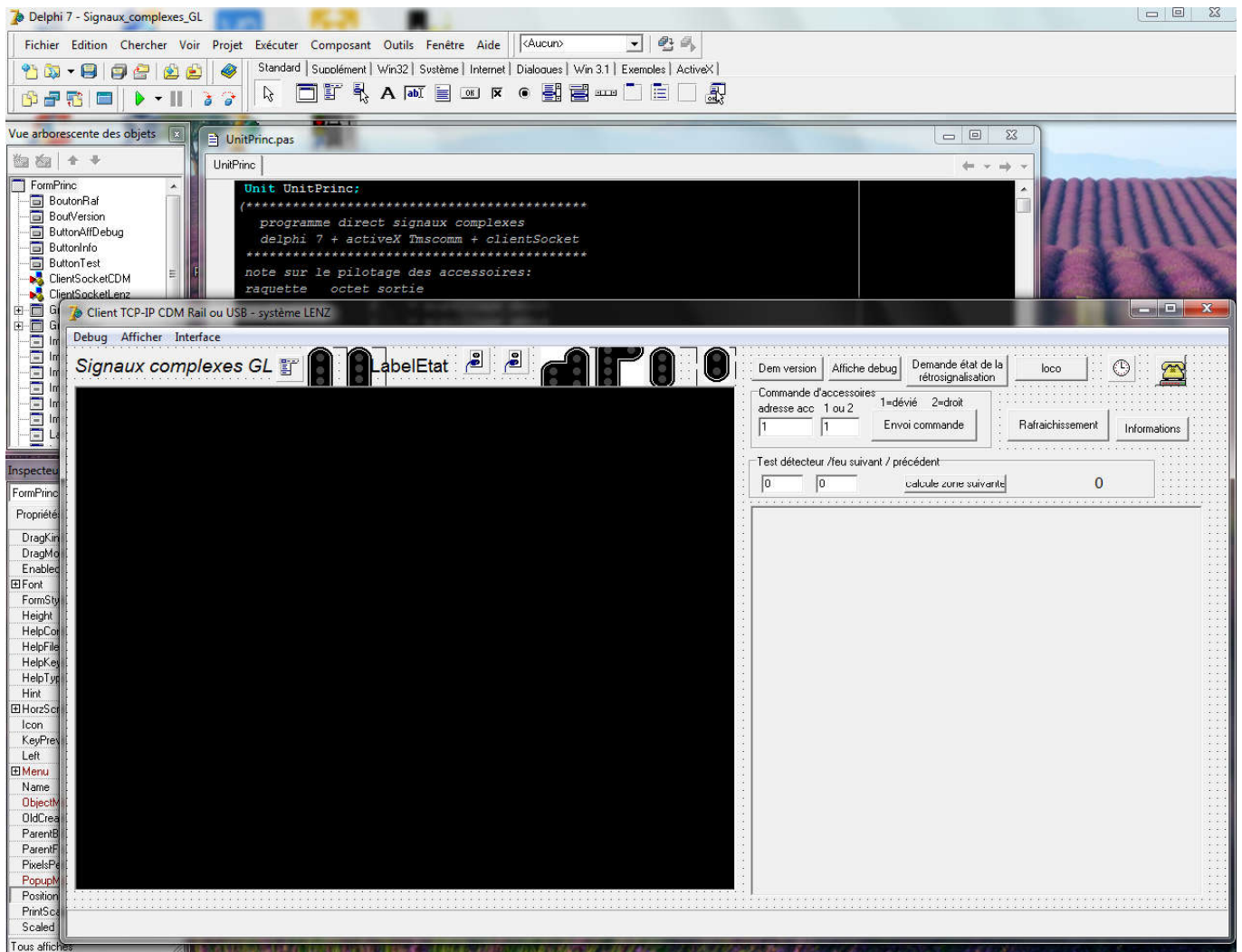
En cas de message « les informations de licence Borland ont été trouvées mais elles ne sont pas valides »

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 :



Généralités sur la détermination des routes sur les évènements détecteurs

En mode connecté à la centrale Lenz, la rétro-signalisation 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 `calcul_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	premier train : route de 531 à 518
531 à 518 => Mem 518 à 523	Le détecteur 518 est le dernier de la ligne précédente, c'est donc le même train, le 1.
526 à 513 => Mem 513 à 531	Le détecteur 513 n'avait pas été engagé dans les lignes précédentes, c'est donc un nouveau train (2)
518 à 523 => Mem 523 à 526	le détecteur 523 était le dernier de la ligne du train 1, c'est donc le train 1.
513 à 531 => Mem 531 à 518	etc
531 à 518 => Mem 518 à 523	
523 à 526 => Mem 526 à 513	
518 à 523 => Mem 523 à 526	

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.

Gestion de la centrale LENZ

En mode autonome, la centrale transmet des informations via le bus XpressNet 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 des 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.

alg= algorithme :

1=arrêt sur l'élément suivant qu'il soit un détecteur ou un aiguillage

2=arrêt sur aiguillage en talon mal positionné

3=arrêt sur un aiguillage pris en pointe dévié ; la variable globale AdrDevie contient l'adresse de l'aiguillage dévié ainsi que son type dans 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

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(518,1,20,2,3) ;`

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);
```


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 codifiées de façon différente des centrales. Les informations sont reformatées comme suit:

Codification de la position des aiguillages

<u>Transmis par CDM</u>	<u>Transmis par le bus XpressNet</u>
Aiguillage simple	
Dévié à gauche = 3	Dévié = 1
Dévié à droite = 1	Droit (non dévié) = 2
Non dévié = 0	
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	
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

La procédure **affiche_signalCplx(int adresse)** aiguille le programme vers le pilotage des décodeurs correspondant aux feux :

```
// pilotage d'un signal
procedure envoi_signal(Adr : integer);
var i : integer;
begin
  i:=index_feu(Adr);
  case feux[i].decodeur of
    0 : envoi_virtuel(Adr);
    1 : envoi_signalBahn(Adr);
    2 : envoi_CDF(Adr);
    3 : envoi_LDT(Adr);
    4 : envoi_LEB(Adr);
    5 : envoi_NMRA(Adr);
    6 : envoi_UniSemaf(Adr);
  end;
end;
```

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 = 2 : aiguillage non dévié
aiguillage[index].position = 1 : aiguillage dévié
```

Informations et procédures relatives aux signaux

`Maj_Etat_Signal(adresse,etat)` : Procédure qui met à jour l'état du signal dans le tableau `EtatSignalCplx` 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, `Etat_signalCplx[316]=10000100000000` en binaire.

A la suite de ces 2 appels, si on lance

`Maj_Etat_Signal(316,ral_30)`, `Etat_signalCplx[316]=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.

`Etat_signalCplx[adresse]` : contient l'état du signal complexe ou du panneau directionnel codé en binaire :

```
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).

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

`ancien_tablo_signalCplx[adresse]` : 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, dessines les états dans
// la fenêtre graphique et dans le TCO.
```

Information sur la vitesse d'un train (uniquement en mode RUN avec train)

`vitesse[adresse]` contient la vitesse actuelle du train dont l'adresse est « adresse ».

Information sur les détecteurs

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

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

`Det_origine[detecteur] = adresse` du détecteur d'où vient le train présent sur le détecteur « détecteur »

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

Information sur les actionneurs (valable uniquement en mode RUN et non TCO)

`Tablo_actionneur[index]` : contient les informations de l'actionneur

Informations et procédures diverses

`Changement` est une variable booléenne qui prend la valeur vraie en cas de changement d'état d'un détecteur, un actionneur ou un signal CDM. Elle est remise à faux en fin de procédure.

`demarre_feu_rouge(INT Signal,INT det_feu)` : permet de démarrer un train 5 secondes après son arrêt devant un signal au rouge. `AdrSignal` : adresse du signal CDM considéré ; `det_feu` : adresse du détecteur de zone du feu (il permet d'identifier le train arrêté devant le signal après le l'entrée en zone)

Liste des structures de données utilisées

Codification des aiguillages :

```
Taiguillage = record
    Adresse : integer;
    modele : TEquipement;
    position, // position actuelle : 1=dévié 2=droit (centrale LENZ)
    posInit, // position d'initialisation 1=dévié 2=droit 9=non positionné
    Adrtriple, // 2eme adresse pour un aiguillage triple
    temps, // temps de pilotage (durée de l'impulsion en x 100 ms)
    inversion : integer; // positionné dans fichier config_gl section_init
    InversionCDM : integer ; // pour les aiguillages déclarés inversés dans CDM, utilisé en
                                mode autonome (paramètre I1)
    vitesse : integer; // vitesse de franchissement de l'aiguillage en position
                                déviée (60 ou 90)
    ADroit : integer ; // (TJD:identifiant extérieur) connecté sur la position droite
                                en talon
    ADroitB : char ; // id de branche pour TJD
    ADevie : integer ; // (TJD:identifiant extérieur) adresse de l'élément connecté
en position déviée
    ADevieB : char; // caractère (D ou S) si aiguillage de l'élément connecté en
position déviée
    APointe : integer; // adresse de l'élément connecté en position droite ;
    APointeB : char;
```

```

DDroit : integer;          // destination de la TJD en position droite
DDroitB : char ;

DDevie : integer;          // destination de la TJD en position déviée
DDevieB : char ;

tjsint  : integer;          // pour TJS
tjsintb : char ;

// éléments connectés sur la branche déviée 2 (cas d'un aiguillage triple)
Adevie2 : integer;
Adevie2B : char ;

// si modifié en mode config
modifie : boolean ;
end;
```

Utilisation :

```

i:=aiguillage[index_Aig(1)].position ; // renvoie la position de l'aiguillage d'adresse 1
adr:=aiguillage[7].Adresse ;          // renvoie l'adresse de l'aiguillage d'index 7
```

structure d'un élément du réseau

```

TBranch = record
    BType : Téquipement ; // ne prend que les valeurs suivantes: det aig Buttoir
    Adresse : integer ;    // adresse du détecteur ou de l'aiguillage
end;
```

structure signaux

```

TFeu = record
    adresse, aspect : integer; // adresse du feu, aspect (2 feux..9 feux 12=direction 2 feux
                                // .. 16=direction 6 feux)
    Img : TImage;              // Pointeur sur structure TImage du feu
    Lbl : TLabel;              // pointeur sur structure TLabel du feu
    check : TCheckBox;          // pointeur sur structure Checkbox "demande feu blanc"
    FeuBlanc : boolean ;        // avec checkbox ou pas
    decodeur : integer;         // type du decodeur
    Adr_det1 : integer;          // adresse du détecteur1 sur lequel il est implanté
    Adr_det2 : integer;          // adresse du détecteur2 sur lequel il est implanté (si un
                                // signal est pour plusieurs voies)

    Adr_det3 : integer;
    Adr_det4 : integer;
    Adr_el_suiv1 : integer;      // adresse de l'élément1 suivant
    Adr_el_suiv2 : integer;
    Adr_el_suiv3 : integer;
    Adr_el_suiv4 : integer;
    Btype_suiv1 : Téquipement ;
    Btype_suiv2 : Téquipement ;
    Btype_suiv3 : Téquipement ;
    Btype_suiv4 : Téquipement ;
    VerrouCarre : boolean ;     // si vrai, le feu se verrouille au carré si pas de train
                                // avant le signal
    modifie : boolean;          // feu modifié
    EtatSignal : word ;         // comme EtatSignalCplx
    UniSemaf : integer ;        // définition supplémentaire de la cible pour les decodeurs
UNISEMAF
    AigDirection : array[1..6] of array of record // pour les signaux directionnels :
contient la liste des aiguillages associés
                                Adresse : integer; // 6 feux max associés à un tableau
                                posAig : char;
                                end;
    CondCarre : array[1..6] of array of record // conditions supplémentaires d'aiguillages
en position pour le carré
                                // attention les données sont stockées en adresse 1 du
tableau dynamique
                                Adresse : integer; // aiguillage
                                posAig : char;
                                end;
end;
```

Procédures et fonctions

Renvoie l'adresse du détecteur suivant des deux éléments

Det1 et Det2 peuvent être séparés par des aiguillages jusque 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 ;
```

Renvoi 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 événements 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;
```

Insertion de code pour un signal spécifique

Pour insérer le code pour un signal spécifique il faut le faire dans la procédure Maj_Feu() :

On y teste l'adresse spécifique des feux (ici 201 et 217) pour y effectuer le traitement, et on sort de la procédure par un return.

```
// mise à jour de l'état d'un feu en fonction de son environnement et affiche le feu
procedure Maj_Feu(Adrfeu : integer);
var i,j,k1,k2,BtypeSuiv,Adr_det,etat,Adr,Aig,DetPrec1,DetPrec2,Detprec3,Detprec4,Adr_El_Suiv,
    Btype_el_suivant,det_initial,bt,el_suiv,modele : integer ;
    PresTrain,Aff_semaphore,car : boolean;
    s : string;
begin
    s:='Traitement du feu '+intToSTR(Adrfeu)+'-----';
    if AffSignal then AfficheDebug(s,clOrange);
    i:=index_feu(Adrfeu);
    if AdrFeu<>0 then
    begin
        modele:=Feux[i].aspect;

        Adr_det:=Feux[i].Adr_det1; // détecteur sur le signal
        Adr_El_Suiv:=Feux[i].Adr_el_suiv1; // adresse élément suivant au feu
        Btype_el_suivant:=Feux[i].Btype_suiv1;

        // signal directionnel ?
        if (modele>10) then
        begin
            //Affiche('Signal directionnel '+IntToSTR(AdrFeu),clyellow);
            Signal_direction(AdrFeu);
            exit;
        end;

        // signal non directionnel
        etat:=etat_signal_suivant(AdrFeu,1) ; // état du signal suivant + adresse du signal suivant dans
Signal_Suivant
        if AffSignal then AfficheDebug('Etat signal suivant ('+intToSTR(AdresseFeuSuivant)+') est
'+intToSTR(etat),clyellow);
```

```
// signaux traités spécifiquement
if (AdrFeu=201) then
begin
    if ((aiguillage[28].position<>const_droit) and (aiguillage[29].position<>const_droit) and
        (aiguillage[31].position=2)) then // attention spécial
        Maj_Etat_Signal(AdrFeu,blanc) else Maj_Etat_Signal(AdrFeu,violet);
    envoi_LEB(AdrFeu);
    exit;
end;
if (AdrFeu=217) then
begin
    if ((aiguillage[24].position<>const_droit) and (aiguillage[26].position<>const_droit)) then
        Maj_Etat_Signal(AdrFeu,blanc) else Maj_Etat_Signal(AdrFeu,violet);
    envoi_LEB(AdrFeu);
    exit;
end;
```



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

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 sont les suivants:

- état des aiguillages
- état des actionneurs
- état des détecteurs

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.

Exemple de programmation du kit client

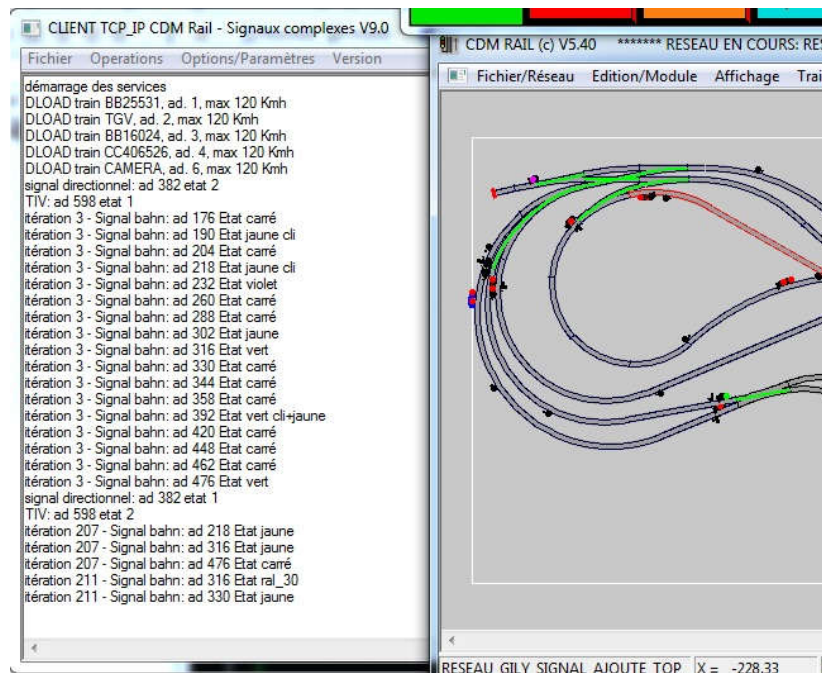
Le kit client est un exemple d'implantation des fonctions de pilotage des signaux complexes. Pour adapter le kit client à votre réseau il faut renseigner le fichier config.cfg de définition des aiguillages et des branches de réseau.

Utilisation du programme client

- Lancer **CDM rail**
- Lancer le serveur d'interface (**interface/démarrer un serveur**)
- Lancer le serveur IP (**comm IP/démarrer le serveur comm IP**).
- Lancer le programme client signaux_complexes. Il doit afficher "client démarré".

Dans CDM rail, passer en mode RUN avec ou sans trains (TCO)

Ci dessous, CDM rail en exécution avec le client « signaux complexes »



L'affichage dans le programme client donne l'état des pilotages des signaux et l'état des mémoires de zone. Il est possible de sélectionner l'affichage de l'un ou de l'autre dans le menu Options/paramètres « afficher évènements feux » et « afficher évènements mémoires de zone » (menus à bascule).

Pilotage d'essai

Le pilotage direct d'un feu peut être réalisé depuis le programme client sans programmation depuis le menu « opérations/commande accessoire ». **Ne pas oublier de démarrer préalablement le serveur IP dans CDM rail :**

- Lancer **CDM rail**
- Lancer le serveur d'interface (**interface/démarrer un serveur**)
- Lancer le serveur IP serveur (**comm IP/démarrer le serveur comm IP**)

Il n'est pas nécessaire d'ouvrir un LAY

Toutes les manipulations suivantes se font depuis le programme client "signaux_complexes" :

- lancer le programme client **signaux_complexes**. Au démarrage, le client doit se connecter au serveur (CDM rail) et doit afficher "client démarré". Attention aux antivirus et aux pare feu qui bloquent les ports de communications IP.
- choisir le menu **opérations/commande accessoires**, une fenêtre s'ouvre:

Dans l'adresse de base, mettre l'adresse du décodeur sur lequel le signal est branché, puis cocher l'adresse dont on veut activer la fonction, puis cocher :

« out1 » pour activer bit 0 (sortie 1) (= rouge programme hex manipu) (= touche + de la LH100 chez lenz)
ou
« out2 » pour activer bit 1 (sortie 2) (= vert programme hex manipu) (= touche - de la LH100 chez lenz)

puis cliquer sur OK.

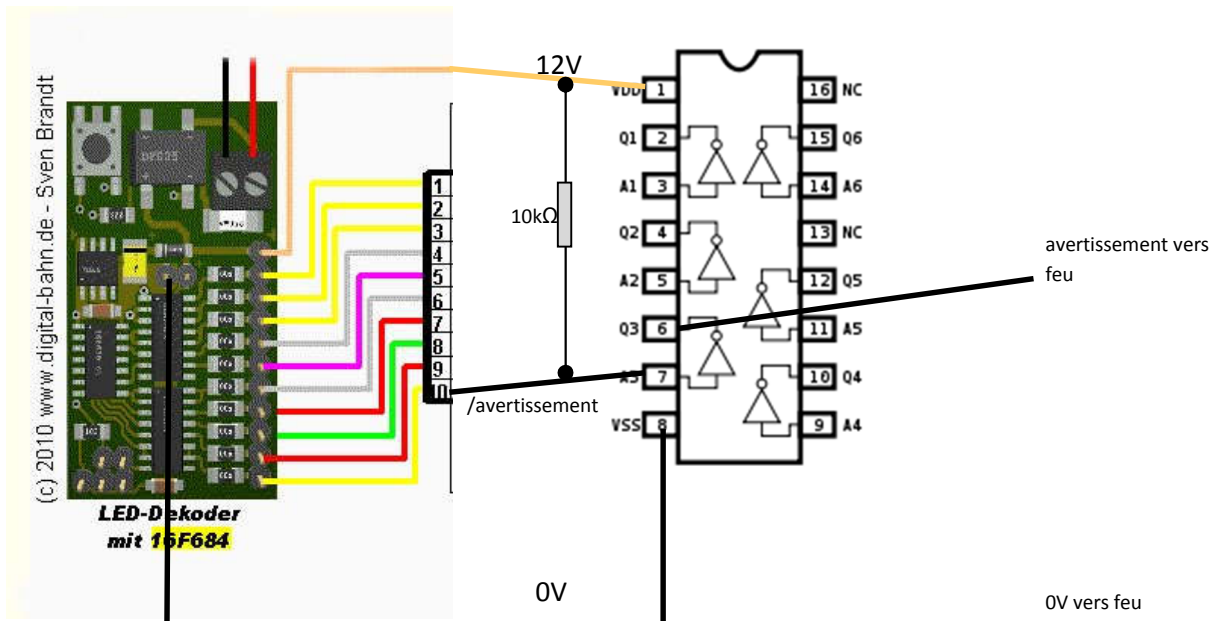
Arrêt des logiciels

Pour l'arrêt des 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 -

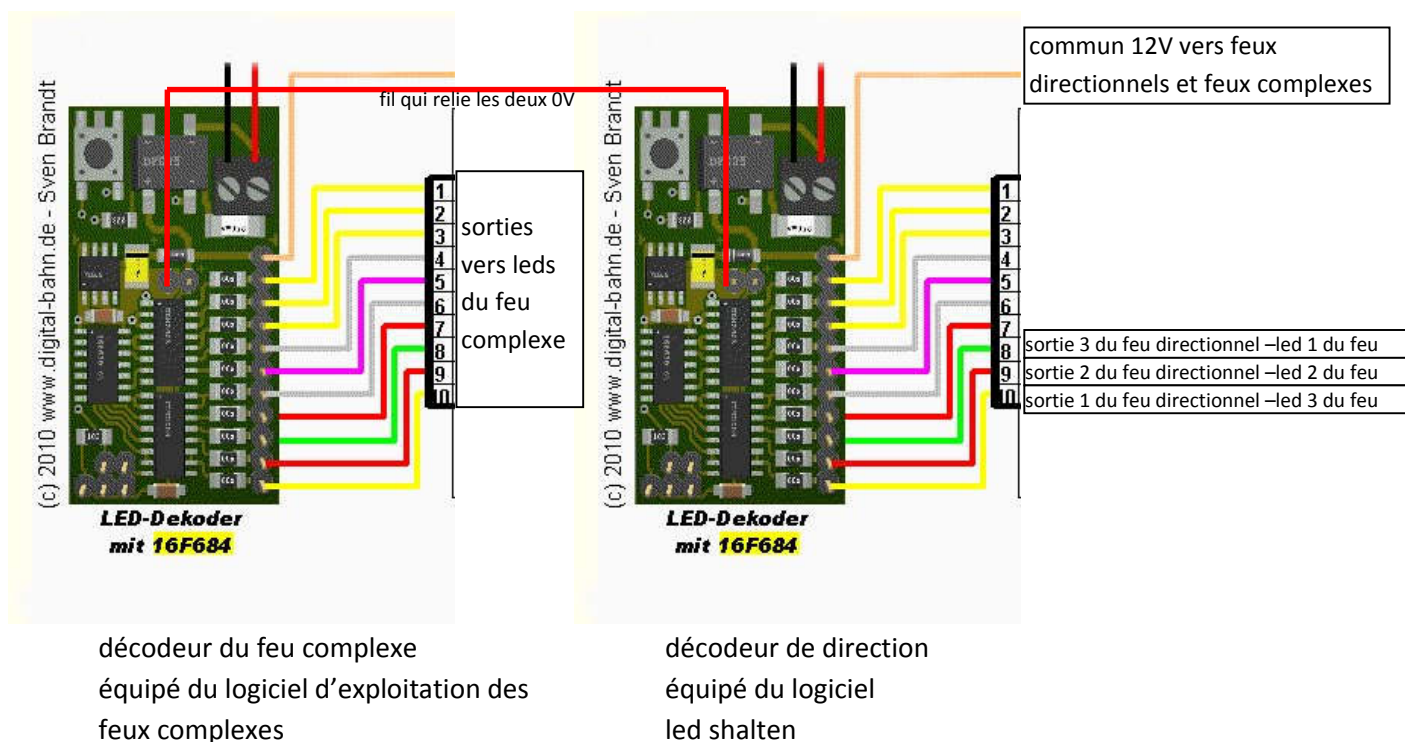
Ci dessous figure le schéma d'une interface permettant l'utilisation de signaux avec un commun au + avec les décodeur digitalbahn avec un circuit CMOS 4049 (6 inverseurs)



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) :



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	
	CDF		

gestion de l'œilleton		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. Sémaphore | arrêt ou franchissable |
| 3. Sémaphore 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 manoeuvre autorisé |
| 8. Blanc clignotant | départ en manoeuvre limitée autorisé |
| 9. Avertissement | annonce un carré, un sémaphore, ou un sémaphore 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 |