Notice d'utilisation du programme signaux_complexes_GL

Cette notice montre comment utiliser le programme signaux_complexes_GL pour utiliser des signaux complexes français avec CDM rail.

Exemples de signaux complexes :

ralentissement

Rappel de ralentissement

carré

Rappel de ralentissement combiné à un avertissement

Matériel nécessaire :

- Le programme « signaux_complexes_GL»
- 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 **UniSemaf**

Introduction

Signaux_complexes_GL est une évolution graphique du programme signaux complexes.
Le programme « signaux complexes » a été écrit en C++ (avec codeblocs), et « Signaux_complexes_GL » a été écrit en Delphi7 enrichi d'un OCX pour la communication série et USB, et de la bibliothèque sockets.

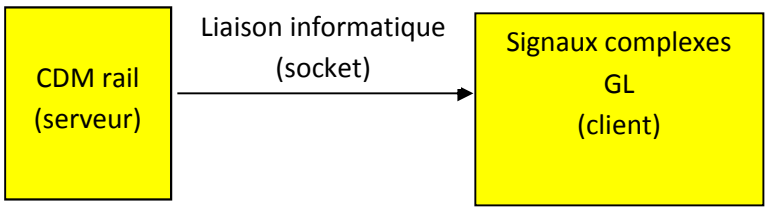
Ils utilisent le même moteur logique de localisation des trains sur le réseau à partir des évènements des détecteurs de zone.

Ces deux programmes utilisent le même fichier de configuration (*config.cfg*) qui sont donc interchangeables. Le programme signaux_complexes_GL utilise un deuxième fichier de configuration (*client_gl.cfg*) qui contient des paramètres qui lui sont spécifiques. Il est décrit en fin de cette notice.

Signaux_complexes_GL a deux modes de fonctionnement :

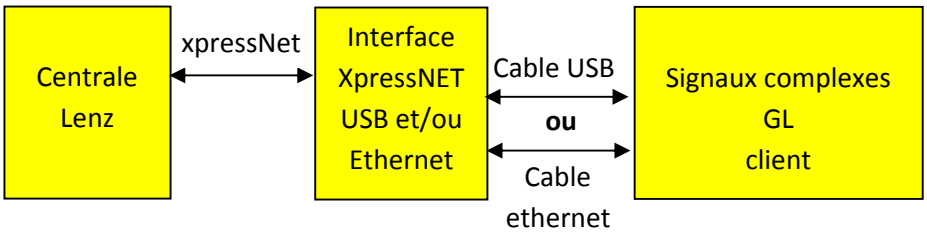
Fonctionnement avec CDM rail et signaux_complexes_GL

Il peut être client de CDM rail pour piloter les signaux complexes. Il peut être exécuté sur le même ordinateur que CDM rail, ou un ordinateur différent. Dans ce cas ces deux ordinateurs seront reliés via réseau, et il est nécessaire de renseigner l'adresse IP V4 du PC exécutant CDM rail dans le fichier client_gl.cfg.



Fonctionnement de Signaux_complexes_GL en autonome

Signaux_complexes_GL peut être autonome. Dans ce cas il n'a pas besoin de CDM rail pour piloter les accessoires du réseau. Cette fonctionnalité est valide uniquement avec les centrales LENZ et compatibles. Dans ce cas, Signaux_complexes_GL doit être relié à la centrale par le bus XpressNet (via l'interface XpressNET-USB ou XpressNet-USB-Ethernet ou Genli). Ce mode permet de piloter les locomotives en mode « raquette », et le programme se charge de piloter les signaux complexes. En aucun cas il ne gère la sécurité des convois.



Il existe un mode hybride, qui permet d'utiliser CDM et le programme Signaux_complexes_GL de façon autonome, c'est-à-dire sans liaison client-serveur entre les deux programmes. Dans ce cas, il faut mettre 0 dans la variable « Adresse IP du PC exécutant CDM rail ». CDM rail communique avec l'interface par la liaison USB et Signaux_complexes_GL communique avec l'interface en Ethernet.

Installation

Il est indispensable de procéder à l'installation du logiciel. Pour cela, faire un clic droit sur le fichier `Install.bat` ou `Install2.bat` et sélectionner « Exécuter en tant qu'administrateur ». Il est possible qu'en cas de présence d'un antivirus sur le PC, il refuse de lancer `Install.bat`. Dans ce cas lancer « `installeur.exe` » toujours en mode administrateur.

L'installation va installer un composant nécessaire à la communication USB avec la centrale ainsi que ses autorisations dans le registre. Signaux_complexes_GL ne pourra pas se lancer si ce composant (mscomm32.ocx) n'est pas installé.

Pour une mise à jour, il n'est plus nécessaire de procéder à une installation, il suffit seulement de copier le fichier exécutable (signaux_complexes_GL.exe) dans votre répertoire de travail.

A l'exécution de Signaux_complexes_GL, si votre PC est connecté à Internet, une vérification d'une nouvelle version est lancée et il vous sera proposé de la télécharger. Si vous choisissez oui, le fichier zip sera réceptionné dans le répertoire téléchargements de votre PC.

Pare feu

Pour pouvoir utiliser la liaison socket entre CDM rail et le logiciel signaux_complexes_GL, si le pare feu windows ou de votre antivirus est activé, il faut leur ajouter une exception pour CDM rail ET signaux_complexes_GL (Dans le pare feu windows : *autoriser une application ou une fonctionnalité via le pare feu windows*), cliquer sur autoriser un autre programme, chercher CDM rail puis dans un deuxième temps, signaux_complexes_GL.exe. Le port à autoriser est le 9999 (par défaut).

Exclusion de sécurité

Il est possible que Windows defender détecte par abus un trojan dans Signaux_complexes_GL.exe. Dans ce cas il faut ajouter une exclusion de sécurité en suivant le procédé décrit à ce lien :

https://support.microsoft.com/fr-fr/help/4028485/windows-10-add-an-exclusion-to-windows-security

Vous pouvez toujours faire scanner le fichier pour le vérifier sur un site antivirus comme *Kaspersky* par exemple :
https://virusdesk.kaspersky.fr/

Refus de modification des fichiers du dossier par windows ou d'éxécution

Il est possible qu'en cas de modification des fichiers de configuration, un refus d'accès soit affiché par windows (W10). Il s'agit d'un problème de droit administrateur. Pour remédier à cela suivre la manipulation suivante à faire en mode administrateur. Faire clic droit sur le répertoire de signaux_complexe_GL, puis sélectionner propriétés et ensuite l'onglet sécurité.

Cliquer sur le bouton modifier au milieu.

Dans la sélection supérieure, choisir le profil (ex *tous les utilisateurs* ou *administrateur* ou le nom de votre session windows).

En bas, cliquer sur la ligne contrôle total puis cliquer sur autoriser et ok puis encore ok. Les autorisations totales ont été déclarées pour les profils choisis.

Vous pouvez appliquer la même règle pour tous les profils.

Fonctionnement

Les signaux gérés de base par CDM rail sont de trois types : carré violet, canton 3 feux ou 4 feux. Néanmoins, CDM rail gère ces feux de la même façon. Pour pouvoir utiliser les feux complexes, il faut utiliser un décodeur externe, piloté par le programme. Ce programme client communique avec le programme serveur (CDM Rail).

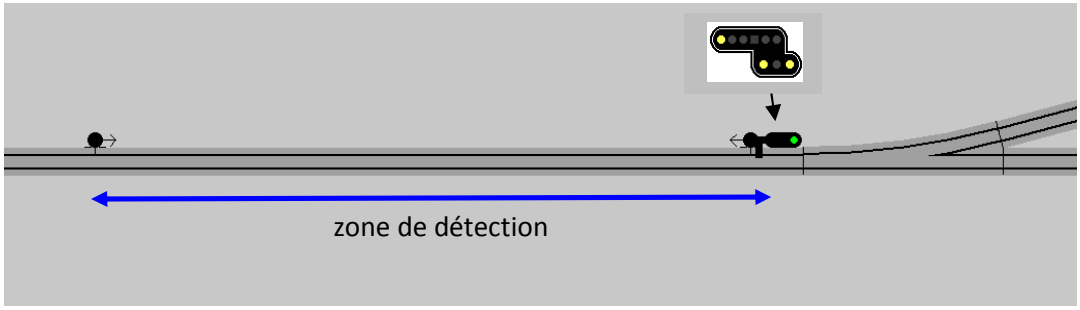
> Les décodeurs des signaux complexes étant pilotés par le même bus que les aiguillages, le plan d'adresses des signaux complexes ne doit pas interférer avec les adresses des aiguillages ou des actionneurs.

Les signaux complexes gérés par le programme client n'ont rien à voir avec les signaux déclarés dans le réseau dans CDM rail. La représentation de la signalisation étant obligatoire sous CDM rail, elle ne correspond pas forcément à des signaux physiques installés ; CDM rail ne pilotant pas la signalisation complexe (étendue). Les deux processus sont donc dissociés.

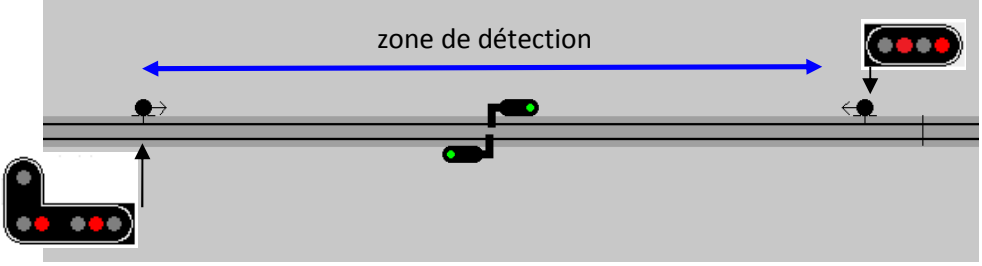
Les signaux complexes du programmes client sont implantables « librement » conformément aux règles de la signalisation française. Ils ne sont pas liés aux signaux de CDM rail. Cela signifie que l'on aura d'une part les signaux implantés dans CDM rail et d'autre part les signaux implantés dans le programme client qui représentent les signaux complexes sur le réseau. A un endroit particulier du réseau on pourra donc avoir un signal CDM et un signal complexe physiquement implanté sur le réseau.

En général on installe les signaux complexes en fin de zone de détection, avant un aiguillage ou un grill pour annoncer le rappel de ralentissement ou le carré (et dans ce cas le signal doit être précédent d'un signal ralentissement).

Exemple :



Un signal complexe est toujours associé à la fin d'une zone de détection (comme ci-dessus), ou ci-dessous dans le cas d'une zone de détection de pleine voie :

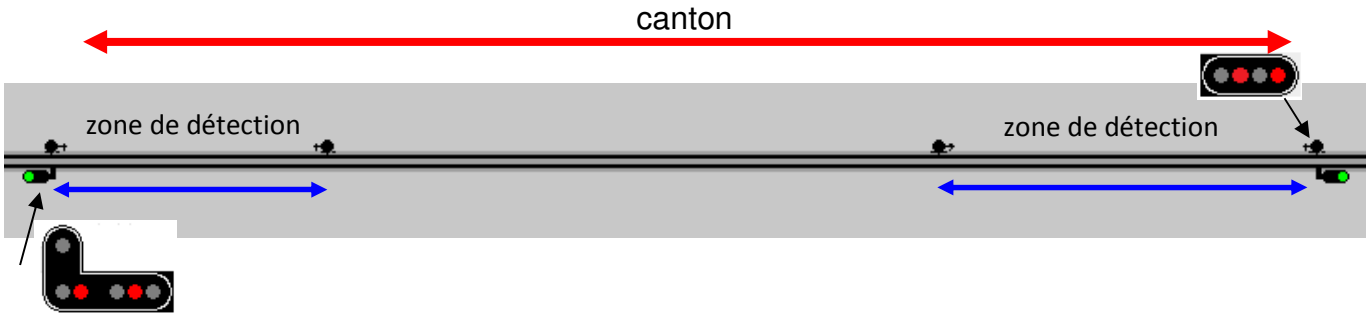


Le signal complexe est bien sûr sensible au sens de circulation de la voie sur lequel il est implanté.

Différences entre cantons et zones de détections

Une zone de détection est un équipement électrique qui permet la détection d'un train sur le réseau.
Un canton peut être constitué de plusieurs zones de détections. Un canton est délimité par des signaux (dits signaux de cantonnement). Deux signaux complexes sont donc positionnés aux limites d'un canton dans lequel un seul train peut être présent. CDM rail conseille de mettre deux zones de détection par canton en regard de chaque signal d'extrémité, mais CDM rail fonctionne aussi avec une seule zone par canton.

Implantation avec deux zones de détection par canton :

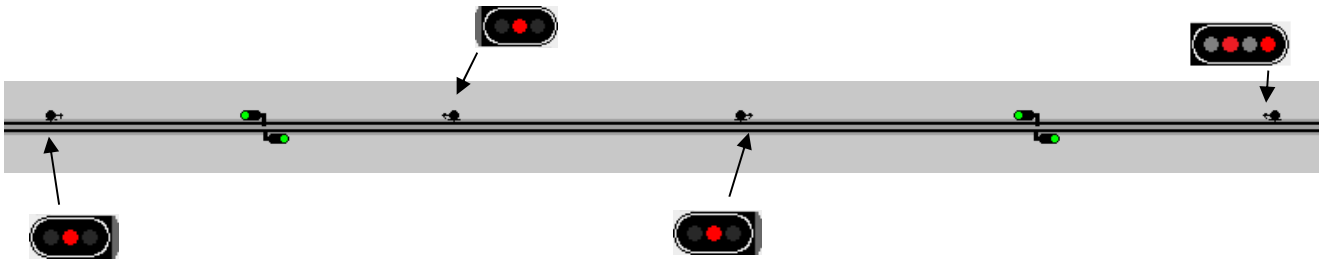


Les signaux complexes sont bien sûr implantés en limite de canton.

Signaux complexes virtuels et réels

Il n'est pas obligatoire *d'implanter* un signal complexe si l'on n'en dispose pas (par exemple pour des questions de coûts). Néanmoins pour des raisons de fonctionnement, il **faut** implanter à la place un signal virtuel (il possèdera une adresse sur le réseau mais ne sera pas piloté). Ceci permet par exemple de piloter un signal de ralentissement malgré que l'on ne soit pas en possession de son homologue, le signal de rappel de ralentissement. En effet le programme signaux_complexes va tester le signal suivant pour connaître son état. Il est donc nécessaire de gérer un signal virtuel. Un signal virtuel a une valeur de décodeur égale à 0 dans le fichier de configuration.

Implantation avec une zone de détection par canton sur des détecteurs de pleine voie :



Pour les signaux complexes, le canton est encadré de deux signaux (cas d'une voie banalisée pour la circulation dans les deux sens) ou d'un seul signal si la circulation se fait toujours dans le même sens.
Les signaux complexes seront implantés sur les fins de zone de détection dans le sens de la circulation, car le changement des signaux complexes se fait sur la retombée de la zone de détection, dans le sens de circulation (front descendant)

Le programme client pilote les signaux complexes du réseau. Il doit donc connaître le réseau. Pour cela, il est nécessaire de renseigner le fichier **config.cfg** qui se trouve dans les fichiers du répertoire du programme de signaux complexes. Il s'agit d'un fichier ASCII modifiable par n'importe quel éditeur de texte (nodepad …)

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.

D'autre part il faut manœuvrer les aiguillages après le passage du train sur le détecteur suivant, sinon la route sera mal évaluée ; elle sera resynchronisée plus tard, mais il aura création d'un train « fantôme » : le nombre de trains affichés sera supérieur.

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 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). Cette opération n'est à faire qu'une seule fois.

Le programme signal client gère les signaux violet, blanc, vert, rouge, carré, sémaphore, jaune, jaune clignotant, ralentissement 30 ou 60 et rappel de ralentissement 30 ou 60.
Pour afficher les signaux vert clignotant, rouge clignotant et blanc clignotant, il faut les programmer spécifiquement le programme par Delphi 7.

Modélisation du réseau, fichier config.cfg

La modélisation du réseau est sa description dans le fichier config.cfg. 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)
Un aiguillage bis est noté avec son adresse suivi de B (exemple A25B)
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 à décrire dans le fichier config.cfg.
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.

Un champ optionnel est noté [ ].

On y décrit un élément par ligne.

```
/*****************************************
/ fichier de configuration de signaux complexes
/ réseau avec signaux complexes
/*****************************************
/ Sans Log=0 / Avec Log=1 : génère un fichier log
Log=0
/ Affichage du débug du calcul des routes, et enregistrement dans le log si la variable précédente est à 1
TraceDet=0
/ si 1  envoie un 0 après le pilotage des décodeurs
RazSignaux=1
/
```

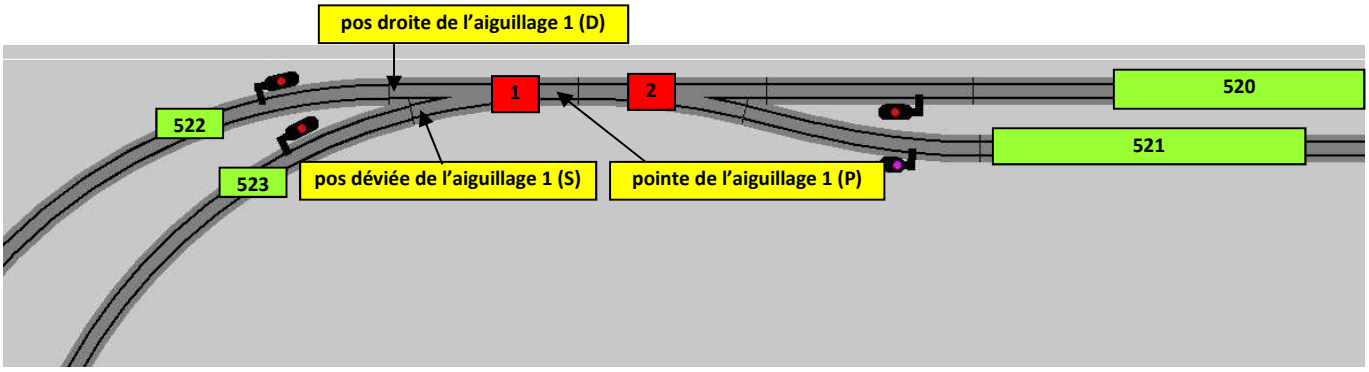
Il y a trois variables au début du fichier de configuration qui constitue la première section :
Log
TraceDet
Ces deux variables ne sont pas utilisées, elles ne sont là que pour assurer l'inter-échangeabilité du fichier avec l'ancien programme « signaux complexes ».
RazSignaux :
Si 1 : envoie une commande 0 après l'écriture des décodeurs de signaux.
Si 0 : n'envoie pas de commande 0 après l'écriture des décodeurs de signaux.

Certains décodeurs autorisent de ne pas envoyer de commande à 0 après l'écriture dans le décodeur de l'aspect du signal, ce qui fait gagner du temps de pilotage des accessoires. Les décodeurs LEB nécessitent la remise à 0 après commande pour qu'ils soient pilotés correctement.
Donc si vous utilisez des décodeurs LEB, il faut mettre cette valeur à 1 ou si vous constatez des pilotages erronés sur le feu du décodeur piloté.

1.Modélisation des aiguillages simples pour la section modélisation des aiguillages

Les aiguillages sont modélisés en décrivant les éléments connectés à leur 3 extrémités.

Exemple 1 : soit l'extrait de réseau suivant, constitué de deux aiguillages d'adresses 1 et 2. Les zones de détection sont en vert et ont pour adresses 520, 521, 522 et 523. Peu importe la distance de l'aiguillage à la zone de détection (ou à un aiguillage), tant qu'elle est contigüe.



Définition des aiguillages : description des éléments connectés sur les 3 extrémités des aiguillages (P= pointe, D=droit, S=dévié) :

Une ligne de la définition d'un aiguillage est constituée des 4 éléments suivants :
Adresse d'aiguillage[B] , **P** (élément connecté à la pointe) , **D** (élément connecté en pos droite) , **S**[2-], (élément connecté en pos déviée) [,vitesse] [,position inversée]

« 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 2ème position déviée à un aiguillage triple (voir plus loin)
[B] est un élément optionnel pour décrire un aiguillage BIS.
[vitesse] est un élément optionnel 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 30 ou 60.

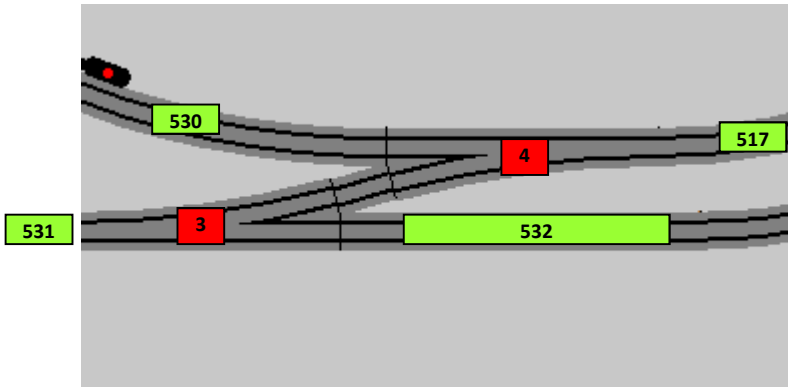
Modélisation des deux aiguillages 1 et 2 ci-dessus :

```
1,P2P,D522,S523
2,P1P,D520,S521,60
```

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**

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

Exemple 2 :



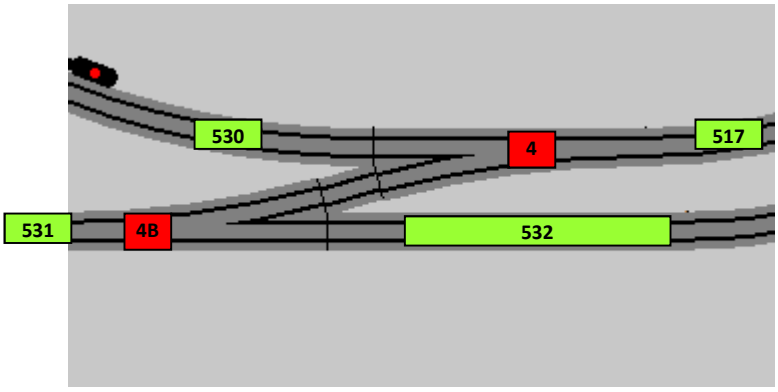
Les aiguillages 3 et 4 auront pour définition :
`3,P531,D532,S4S`
`4,P517,D530,S3S`

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).

Modélisation d'aiguillages ayant la même adresse (aiguillages BIS)



Les aiguillages ci-dessus ont la même adresse dans CDM (4). Néanmoins, pour pouvoir les distinguer dans le programme client, l'un des deux doit recevoir un suffixe « B » (comme BIS). Celui-ci devra toujours être identifié comme BIS. Attention de ne pas inverser un aiguillage BIS avec son homologue non bis dans la section branches ou la section description des aiguillages.

L'écriture des lignes de modélisation pour l'exemple ci-dessus est la suivante :

```
4B,P531,D532,S4S
4,P517,D530,S4BS
```

Restriction importante sur l'utilisation des aiguillages à la même adresse :

A cause d'une erreur dans CDM rail, la connaissance correcte des positions des deux aiguillages renvoyées par CDM vers le programme client est **erronée**. Un sujet est ouvert ici à ce sujet :

http://cdmrail.free.fr/ForumCDR/viewtopic.php?f=13&t=3958

Il n'est donc **pas recommandé** d'utiliser deux aiguillages en bretelle à la même adresse. Il faut les mettre sur deux adresses différentes.

Position inversée

Dans la ligne de définition des aiguillages, il est possible d'indiquer une variable d'inversion de la position :

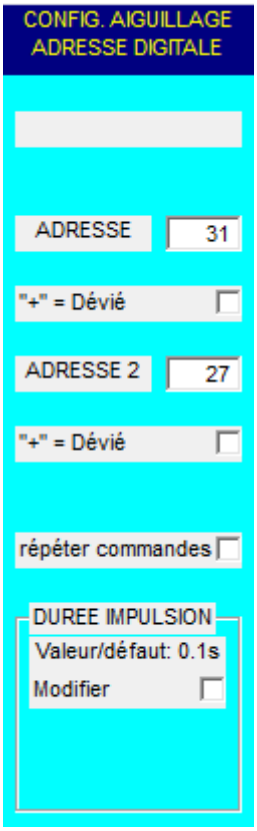
```
4B,P531,D532,S4S,0,1
```

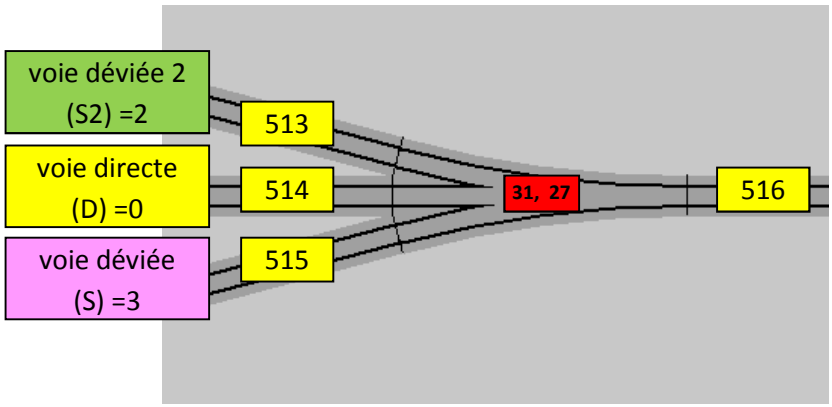
Le 0 représente la vitesse de franchissement

Le 1 représente l'inversion de la position dans le cas ou l'aiguillage BIS est déclaré en commande inversée dans CDM rail. Cette déclaration ne fonctionne que pour les aiguillages BIS.

Modélisation des aiguillages triples

Sous CDM-rail, un aiguillage triple est configuré de la façon suivante :





31 / 27	D	S
D	voie directe (0)	voie déviée (3)
S	voie déviée 2 (2)	-

D=droit S=dévié

Dans l'exemple ci-dessus, l'aiguillage triple comporte les adresses 31 et 27. Cet aiguillage est « connecté » à des détecteurs (513, 514, 515 et 516)

adresse1 TRI, adresse2,
D élement connecté sur la voie directe,
S élément connecté sur la voie déviée (la voie gauche),
S2- élément connecté sur la voie déviée2 (la voie droite),
P élément connecté sur la pointe de l'aiguillage triple

Les éléments D, S, S2 et P peuvent être énumérés dans n'importe quel ordre. Par contre, l'ordre de déclaration de la 1ère adresse et de la 2ème adresse doit respecter le même ordre de déclaration que dans le tableau bleu de CDM-Rail.

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

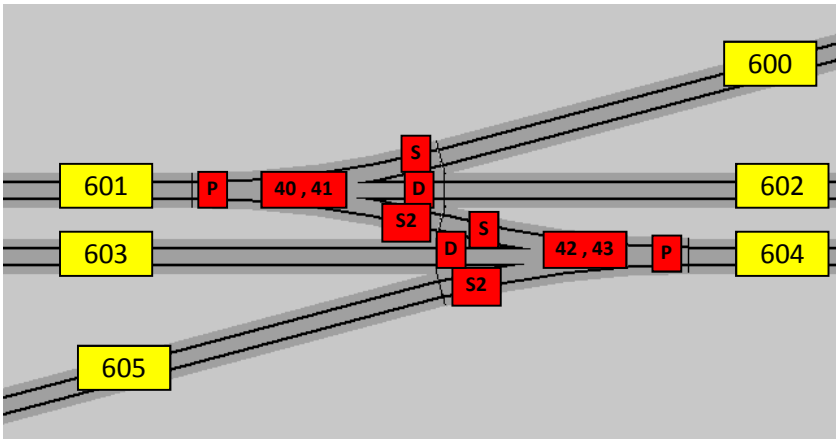
```
31TRI,27,D514,S515,S2-513,P516
```

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
```

La voie droite d'un l'aiguillage triple est toujours la voie déviée 2
La voie gauche d'un l'aiguillage triple est toujours la voie déviée « normale »

Exemple de modélisation de deux d'aiguillages triples



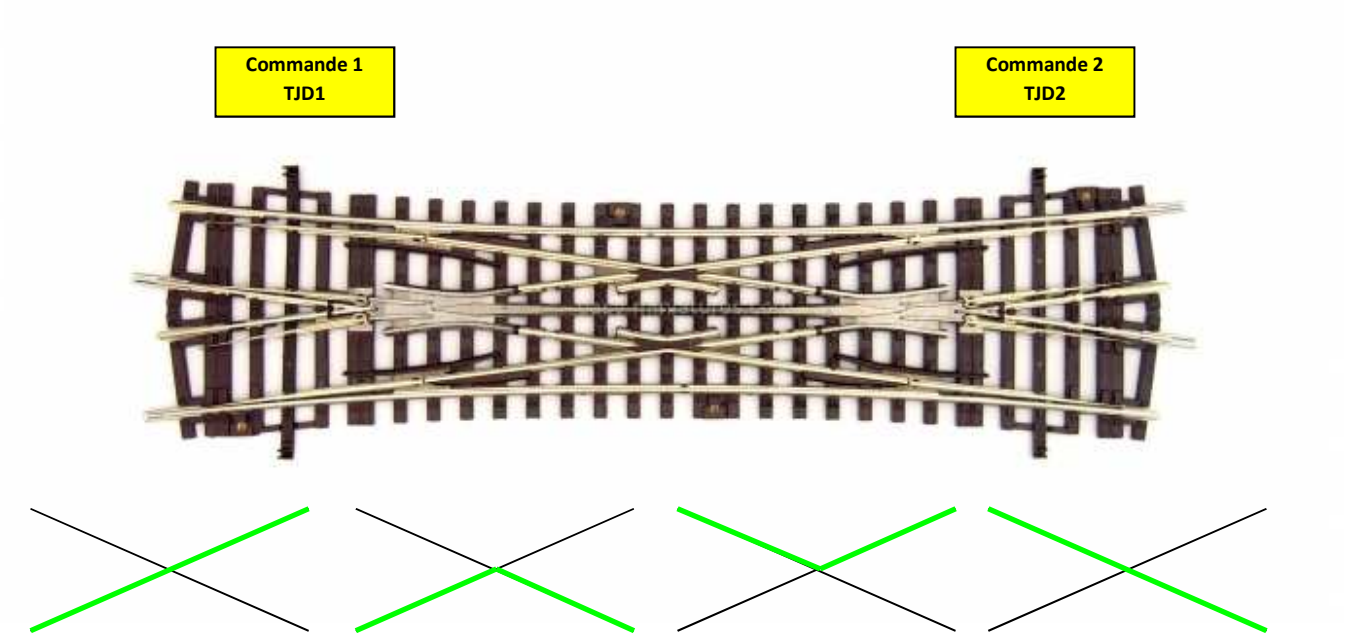
On a ici deux aiguillages triples reliés. Ces deux aiguillages triples sont modélisés comme suit :

```
40TRI,41,D602,S600,S2-42S,P601
42TRI,43,D603,S605,S2-605,P604
```

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

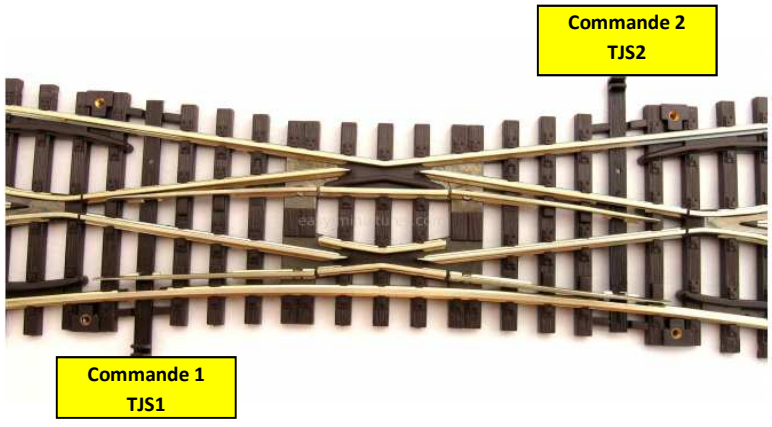
Modélisation des TJD et des TJS

Les TJD comportent 4 aiguilles pour 4 directions possibles :

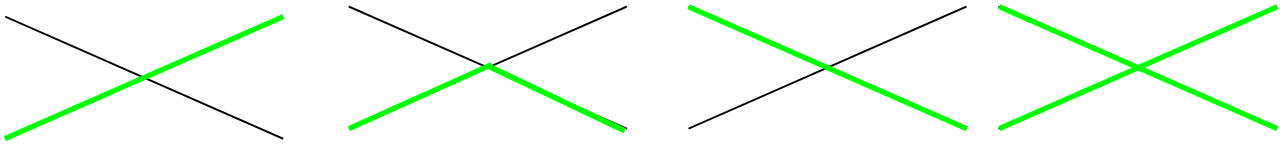


TJD1 déviée TJD2 déviée	TJD1 droite TJD2 déviée	TJD1 déviée TJD2 droite	TJD1 droite TJD2 droite

Les TJS comportent 2 aiguilles pour 3 directions possibles :

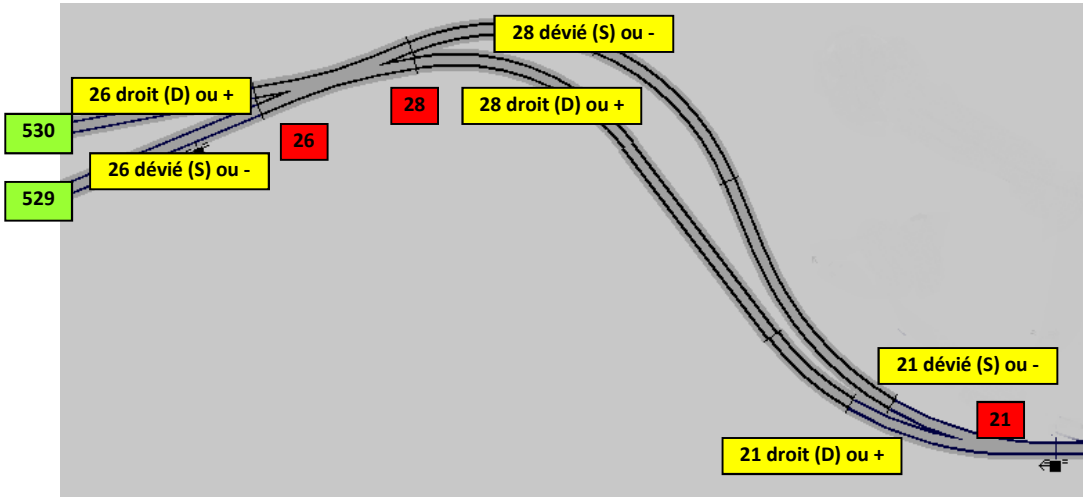


Directions possibles :



TJS1 déviée	TJS1 droite	TJS1 droit	TJS1 déviée

TJS2 déviée TJS2 déviée TJS2 droite TJS2 droite



Une TJD dispose de deux adresses. Pour chaque adresse, on renseigne les éléments connectés en position droite (D) et déviée (S). **Une TJD occupe 2 lignes** dans le fichier config.cfg.

notation :
adresse1_TJD, D élément connecté en droit, S élément connecté en dévié, P 2ème adresse de la TJD
adresse2_TJD, D élément connecté en droit, S élément connecté en dévié, P 1ère adresse de la TJD

Modélisation selon implantation ci-dessus :

```
26TJD,D530,S529,P28
28TJD,D21D,S21S,P26
```

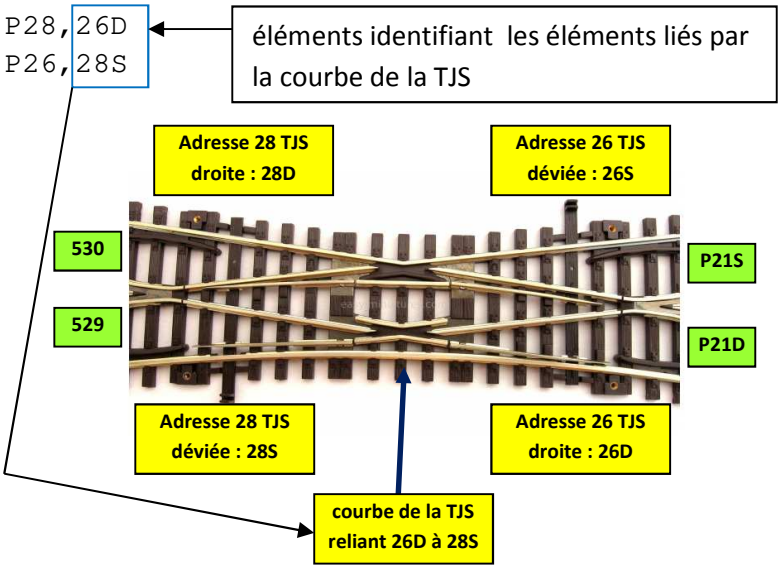
ligne 1
26TJD signifie que l'élément à l'adresse 26 est une TJD.
D530= signifie : l'élément connecté à la position droite de la TJD est un détecteur dont l'adresse est 530.
S529= signifie : l'élément connecté à la position déviée de la TJD est un détecteur dont l'adresse est 529.
P28 = signifie que la 2ème adresse de la TJD est 28.

ligne 2
28TJD signifie que l'élément à l'adresse 28 est une TJD.
D21D= signifie : l'élément connecté à la position droite de la TJD est un aiguillage dont l'adresse est 21 et en position droite.
D21S= signifie : l'élément connecté à la position déviée de la TJD est un aiguillage dont l'adresse est 21 en position déviée.
P26 = signifie que la 1ère adresse de la TJD est 26.

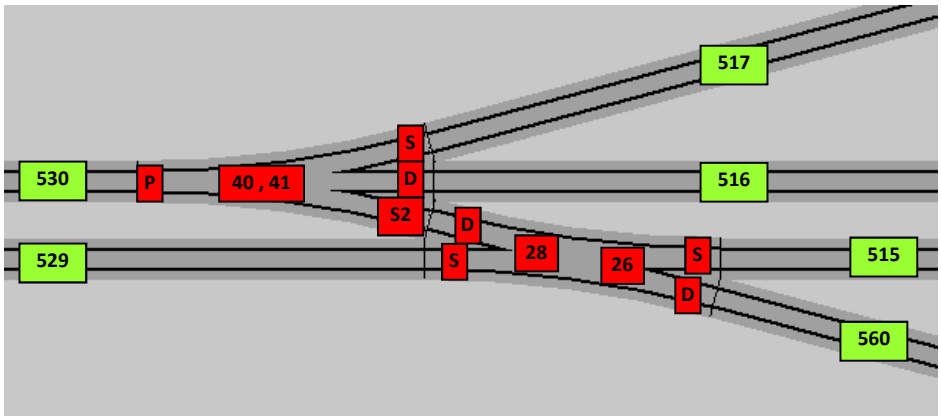
Modélisation d'une TJS

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

```
26TJS,D530,S529,P28,26D
28TJS,D21D,S21S,P26,28S
```

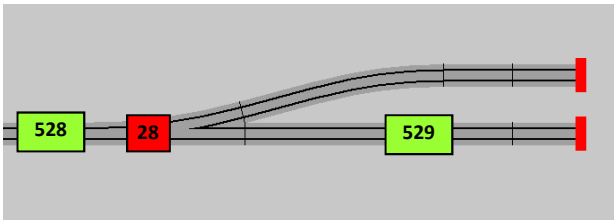


Exemple de modélisation d'un aiguillage triple et d'une TJD




```
40TRI,41,D516,S517,S2-28D,P530
26TJD,D560,S515,P28
28TJD,D40S2,S529,P26
```

Modélisation d'un buttoir



Buttoir supérieur sans détecteur, buttoir inférieur avec détecteur.

L'aiguillage 28 est modélisé comme suit :

`28,D529,P528,S0` L'adresse 0 modélise un buttoir, connecté sur l'élément dévié (S) de l'aiguillage 28.

Dans les branches, le parcours 528 à 529 est modélisé comme suit :

`...,528,A28,529,0` Le 0 terminal modélise un buttoir.

La description du parcours de 28S vers le buttoir n'est pas nécessaire.

L'élément 0 permet également de limiter le parcours à ce point. Par exemple, si la suite du parcours ne comporte plus de détecteurs ou d'aiguillages disposant d'une adresse (aiguillages strictement manuels).

2. Modélisation des branches du réseau pour la section de modélisation des branches

La 2ème partie du fichier config.cfg concerne la modélisation des branches.
Une ligne constitue une branche du réseau.
On y décrit les détecteurs et les aiguillages. Les détecteurs sont renseignés par leur adresse. Les aiguillages sont renseignés par la lettre A suivi de l'adresse de l'aiguillage (suivi de B s'il s'agit d'un aiguillage BIS).
Le sens de description des branches est arbitraire.

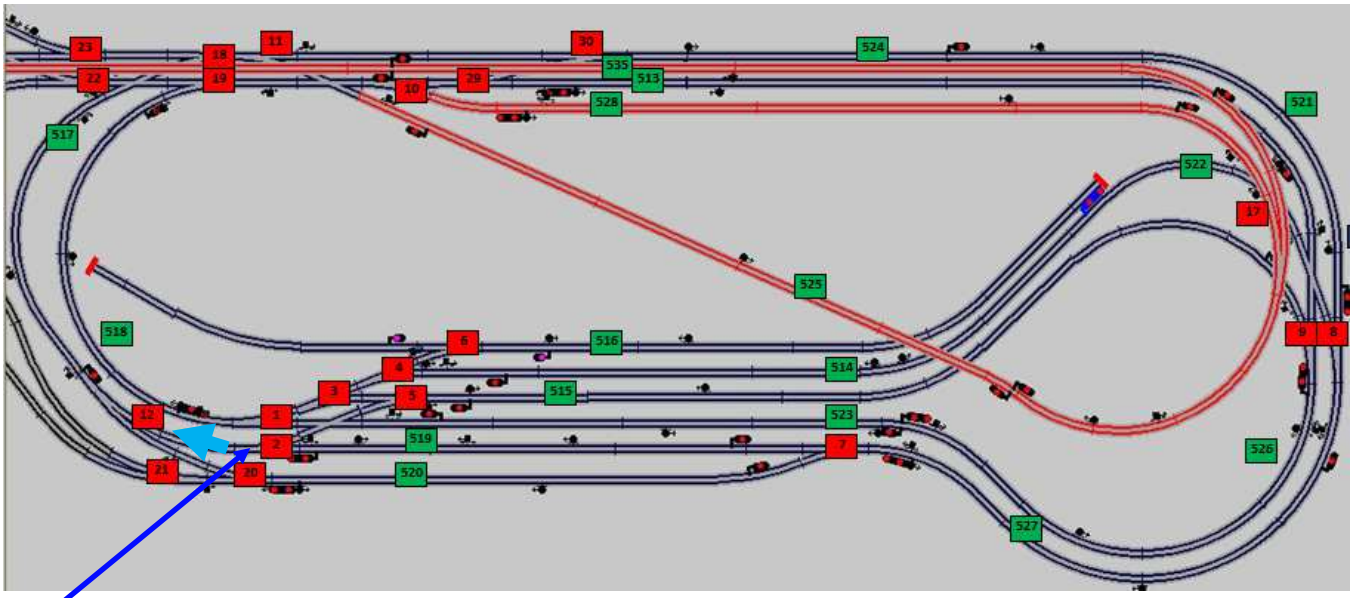
Une ligne doit commencer par un aiguillage (ou un buttoir), contenir au moins un détecteur et se terminer par un aiguillage (ou un buttoir).

Un aiguillage peut se retrouver à plusieurs endroits de cette section. Un détecteur non.

Il y a bien sûr plusieurs façons de réaliser une description de branches, notamment sur l'endroit d'où on commence pour décrire les boucles de même que le sens de parcours.

La liste doit être terminée par 0.

Exemple :




```
A2,A12,517,A18,A11,A30,524,521,A8,527,A7,519,A2
A7,520,A20,A12
A1,A3,A4,514,522,A8
A1,523,526,A9,513,A29,A10,A19,518,A1
A9,515,A5
A11,525,A17,528,A10
A17,535,533,A24,538,A23
A7,520,A20,A21,A28,A26,530,A27,A25,A31,534,A23,A18
A26,529,A25
A22,537,A27
A6,516,0
0
```

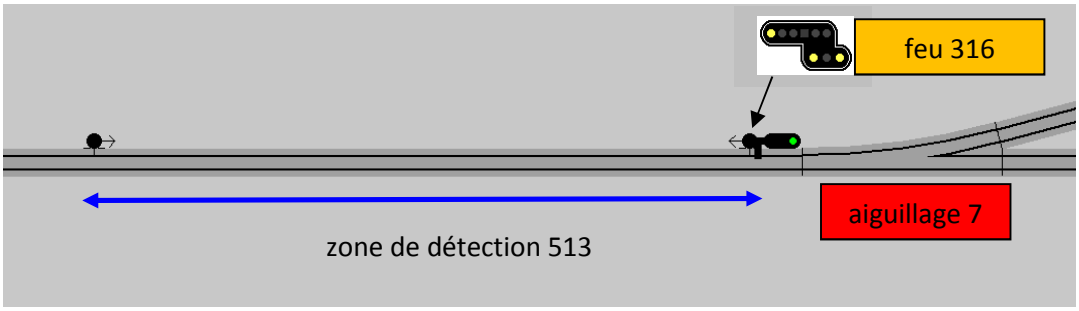
La première ligne est une boucle (de **A2** à **A2**), elle décrit l'anneau extérieur du réseau. On part de l'aiguillage A2 désigné par une flèche. L'élément suivant est l'aiguillage 12 (A12). On ne décrit pas les pointes ou positions déviées ou droite de l'aiguillage car cela a déjà été fait dans la section de description des aiguillages

La deuxième ligne est une branche non bouclée. On part de l'aiguillage A7 en bas jusqu'à l'aiguillage A12 en bas à gauche.

La dernière ligne est un parcours vers le buttoir, elle est donc terminée par 0.

3. 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 , [unisemaf]

forme du signal :

2 = 2 feux (violet blanc) 

3 = 3 feux (rouge jaune vert) 

4 = 4 feux (carré) 

5 = 5 feux (carré + blanc) 

7= 7 feux (blanc + ralentissement) 

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

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

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

réserve : fonctionnalité réservée au feu blanc.

décodeur :
0 = feu virtuel (correspond à un feu géré mais non implanté et non piloté)
1 = Digital bahn
2 = CDF
3 = LDT
4 = LEB
5 = NMRA en protocole DCC étendu (non testé)
6 = Unisemaf de Paco. Ce décodeur nécessite un paramètre supplémentaire dans la ligne, voir le paragraphe spécifique plus loin.

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

Verrouillable au carré :
Cette variable ne concerne que les feux dont la forme est supérieure ou égale à 4.
0 = signal non verrouillable au carré : le signal n'affichera pas de carré si aucun train n'est en approche sur les 3 cantons le précédent.
1 = signal verrouillable au carré si aucun train n'est en approche sur les 3 cantons le précédant.

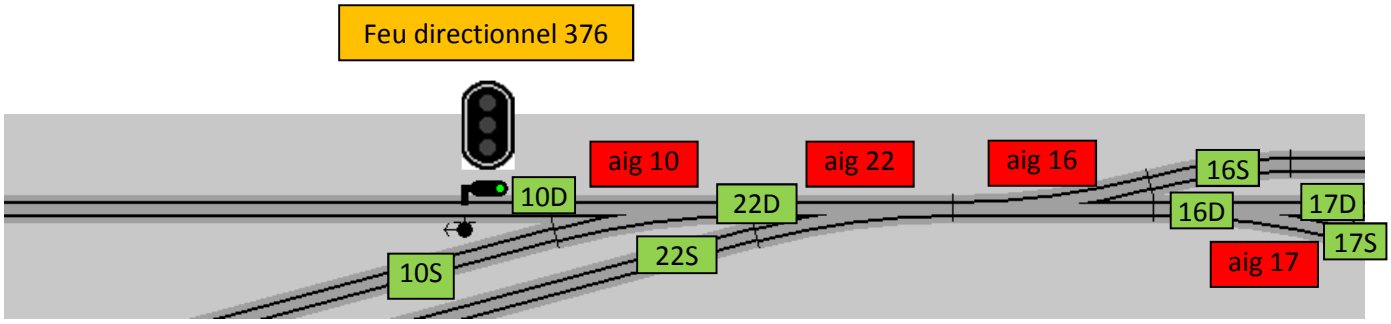
Exemple de déclaration pour le signal ci – dessus :

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.

Signaux directionnels



Adresse du feu, D Nombre de directions, (Aiguillages mal positionnés en condition ou) (aiguillages direction la plus à gauche séparés par une virgule en condition et)
(aiguillages direction 2 séparés par , en condition et) …(aiguillages direction la plus à droite séparés par , en condition et)
Un feu directionnel à 3 feux aura 4 descriptions entre parenthèses. Ils ne sont pas liés à un détecteur.

Exemple pour un signal directionnel (D) à 3 directions d'adresse 376 piloté par un décodeur digitalBahn (1) :

```
372,D3,1,(A10S,A22S)(A16S)(A16D,A17D)(A16D,A17S)
```

0 feu 1 feu 2 feux 3 feux

Cette ligne signifie :
Le feu à l'adresse 372 est un feu directionnel (D) piloté par un décodeur digitalBahn (1).
N'afficher aucun feu si l'aiguillage 10 est dévié (S) **ou** si l'aiguillage 22 est dévié (S) (car dans ce cas ils sont pris en talon et mal positionnés) 

Afficher 1 feu si l'aiguillage 16 est dévié (S) 

Afficher 2 feux si l'aiguillage 16 est droit (D) **et** si l'aiguillage 17 est droit (D) 

Afficher 3 feux si l'aiguillage 16 est droit (D) **et** si l'aiguillage 17 est dévié (S) 

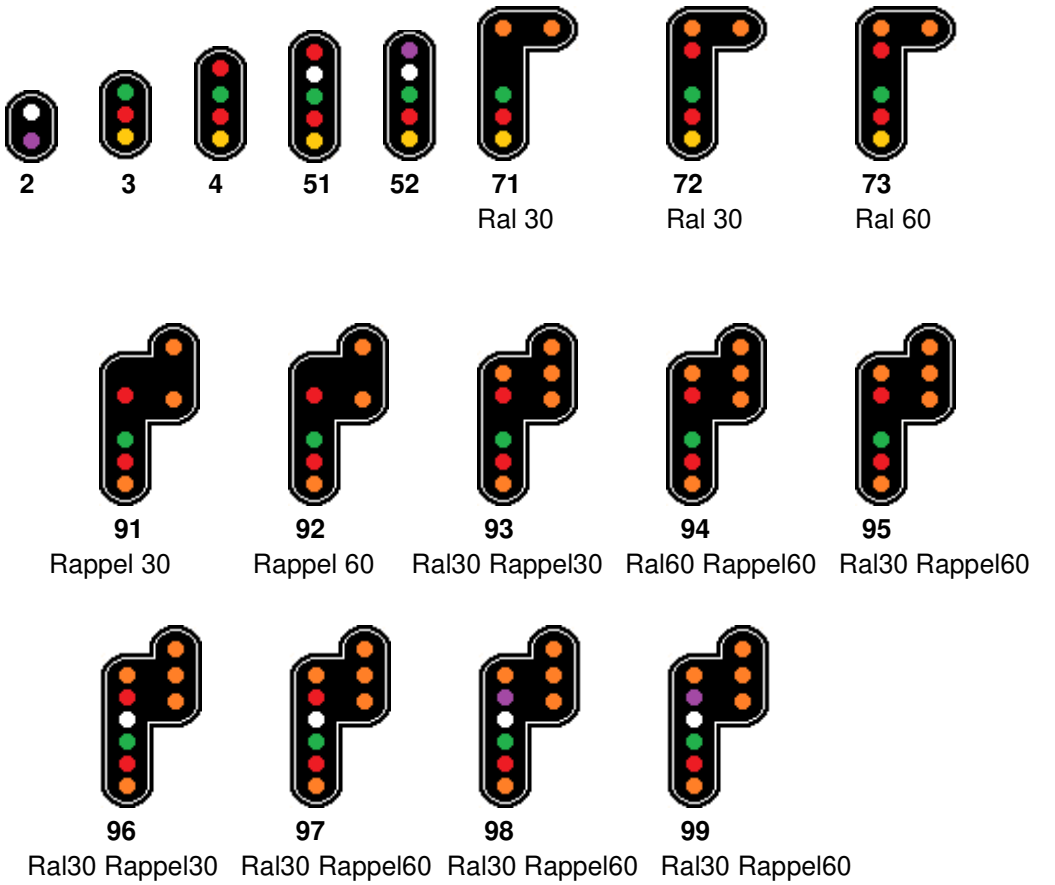
Le nombre d'aiguillages décrits dans une parenthèse est illimité.

Spécificités du décodeur Unisemaf

La ligne de déclaration de ce décodeur nécessite un paramètre supplémentaire qui décrit précisément la cible. Exemple :

`610,7,0,6,(520,A20),0,`**`71`**

Le paramètre supplémentaire (71) est le dernier. Il décrit la cible suivant le tableau ci-dessous (numéro inscrit sous la cible). De ce fait, le 2ème paramètre (ici 7) décrit uniquement le nombre de feux de la cible.



Ce décodeur se paramètre en modifiant ses variables de configuration (CV). Il faut brancher les signaux PQ (de la voie de programmation) sur le bornier X3.

Pour les possesseurs de centrale Lenz :
L'écriture des CV par le mode 8 de la raquette affiche ERR2 mais le CV est quand même écrit.

L'adresse du décodeur sur le bus DCC est stockée dans les CV1 et CV9 selon les règles et formules suivantes :

CV1 de 0 à 63 ; CV9 de 0 à 7

Dcc est l'adresse du décodeur sur le bus Dcc

$$CV1 = \frac{DCC-1}{4} + 1 \text{ modulo } 64 \qquad CV9 = \left(\frac{DCC-1}{4} + 1 \right) / 64$$

Modulo 64 est le reste de la division par 64 / 64 est la division entière par 64

Inversement :

$$DCC = 4\times(64\times CV9 + CV1 - 1) + 1 \qquad DCC = 4.(64.CV9 + CV1 - 1) + 1$$

Exemple : Dcc = 281 CV1 = 7et CV9 = 1

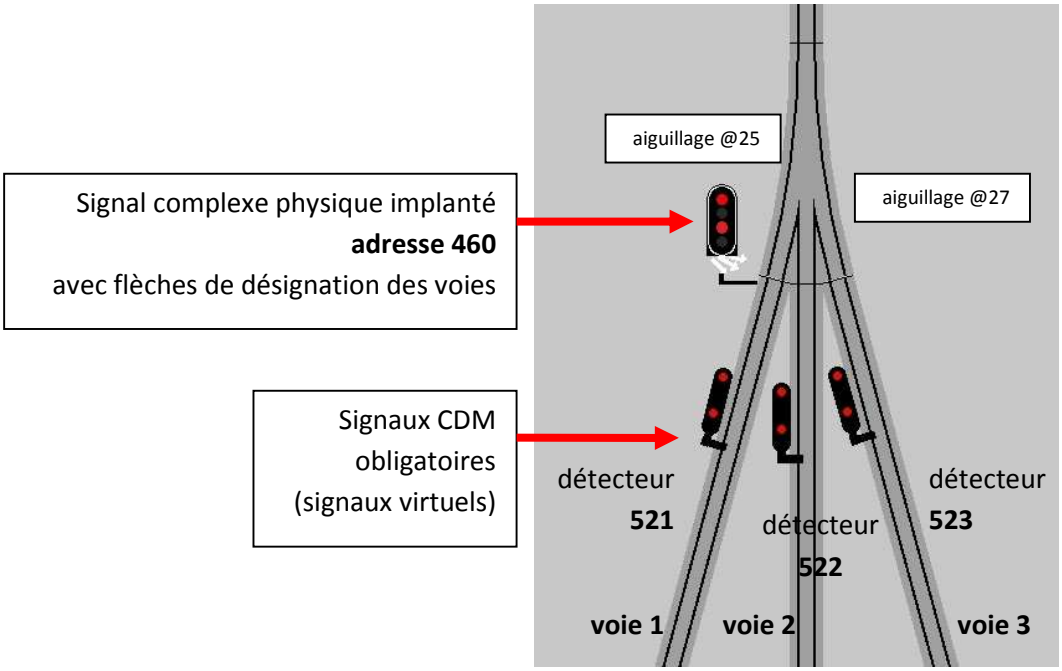
Il est possible d'affecter l'adresse du décodeur par apprentissage en appuyant sur son bouton, et ensuite envoyer à l'adresse désirée une commande d'accessoire avec le décodeur branché sur le bus Dcc. Attention l'adresse envoyée doit être modulo 4. La première adresse du décodeur ne peut prendre que l'une des adresses suivantes : 1 5 9 13 17 21 25 29 … 2033 2037 2041. Si on envoie une adresse différente, l'adresse du décodeur sera ramenée à l'adresse modulo 4 inférieure.

Le changement d'aspect de ce décodeur est obtenu selon le tableau PACO – SNCF établi par Laurent Rieffel. (Doc Unisemaf648). Les CV à partir de 35 doivent être programmés selon ce tableau. Exemple pour afficher un carré sur un signal 51, il suffit d'envoyer une commande + (2) à l'adresse de base du décodeur +1 soit 52.

http://usuaris.tinet.cat/fmco/download/UniSemaf648_manual.pdf

Feux pour plusieurs voies simultanées

Cette configuration permet d'économiser des signaux sur des voies convergentes.



Le signal complexe indique la voie sur lequel l'aiguillage est positionné. Ce signal complexe doit comporter les 3 flèches de désignation des voies, signifiant qu'il implique les 3 voies. Cette configuration est utilisée pour économiser les signaux, notamment sur des faisceaux convergents en gare ou dépôts. Cette configuration est limitée à 4 voies maximum.

modélisation :

```
460,4,0,1,(521,A25,522,A27,523,A27),1
```

Pour l'aiguillage triple, son adresse 25 concerne directement le détecteur 521. L'adresse 27 de l'aiguillage triple concerne les détecteurs 522 et 523.

Exemple de section de signaux dans le fichier de configuration :

```
/ liste des signaux
/ la liste doit être terminée par une adresse à 0
/ forme : 2=2 feux(carré violet/blanc) / 3=3 feux / 4=4 feux (carré) / 5=5 feux (carré + blanc)
/         7=7 feux (carré+blanc + ralentissement  / 9=9 feux (blanc ou violet + rappel ralentissement)
/ type de décodeur : 0=signal virtuel 1=digital Bahn 2=CDF 3=LDT 4=LEB
/ Notation de chaque ligne:
/ adresse de base du signal, forme, réserve, type de décodeur [, (détecteur,..detecteur , élément suivant ..) ,
/ avec ou sans demande de verrouillage du feu au carré (0 ou 1)]
161,4,0,4,(538,A32),1
169,9,0,4,(539,A30),1
177,9,0,4,(569,A23),1
185,4,0,4,(570,A25),1
```

```
193,4,0,4,(516,A29),1
201,2,0,4,(517,31TRI,518,31TRI),0
209,9,0,4,(561,547),0
217,2,0,4,(514,A26,515,A26),0
225,9,0,4,(516,A29),0
233,4,0,4,(547,A20),0
1001,3,0,0,(537,554),0
1003,3,0,0,(553,A1B),0
1005,3,0,0,(571,553),0
1007,3,0,0,(554,A4),0
1009,3,0,0,(522,539),0
1011,3,0,0,(521,569),0
0
```

Les signaux 1001 à 1011 sont des signaux virtuels car leur décodeur est 0. (4ème champ) de chaque ligne.

4. Section Actionneurs

Cette section décrit les actions devant être déclenchés lorsqu'un train active ou désactive 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 F16)

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.
```

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.

Actionner un passage à niveau à une ou plusieurs voies

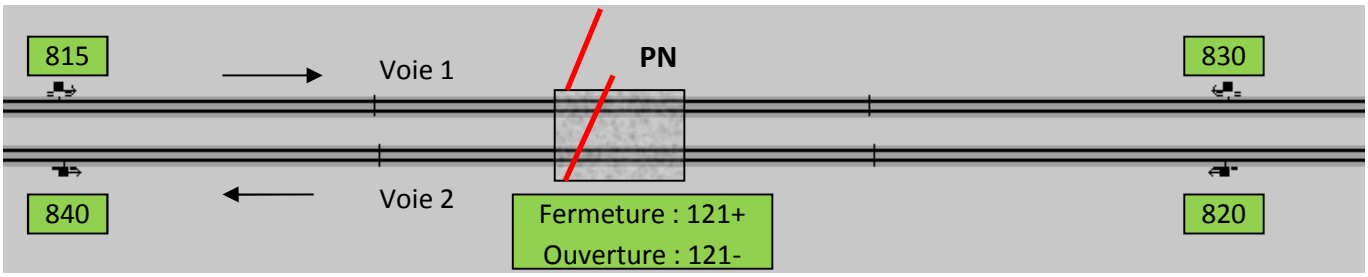
Le déclenchement de l'action « PN » est provoqué par n'importe quel train.

```
(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+-)
```

Le nombre d'adresses des actionneurs de fermeture et d'ouverture dépendent du nombre de voies que traversent le PN. Un passage à niveau à voie unique ne dispose que d'un seul détecteur d'ouverture et un seul détecteur de fermeture.

Action :
PN=fonctionnement pour un passage à niveau

Exemple passage de deux voies sur un PN :
`(815,830),(820,840),PN(121+,121-)`



L'ouverture du PN est provoquée par la montée de l'actionneur 815 ou 820 et font monter une mémoire de présence train sur chaque voie sur laquelle le détecteur 815 ou 820 a été activé.
La mémoire de présence train repassera à 0 après que les deux actionneurs 830 et/ou 840 se désactivent (queue du train).

Le sens de détection des actionneurs suit le sens de circulation de chaque voie.

Le passage d'un train sur la retombée de l'actionneur 830 et/ou 840 feront retomber la mémoire de présence train et le passage à niveau s'ouvrira (la zone des **deux** voies devra être dégagée).
La fermeture du PN est exécutée en envoyant + à l'adresse DCC 121.
L'ouverture du PN est exécutée en envoyant – à l'adresse DCC 121.

Si d'autres voies traversent le PN, il suffit de les ajouter à la suite avant la commande PN:

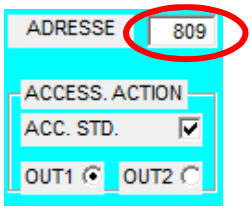
```
(815,830),(820,840),(816,831),PN(121+,121-)
```

Voie 1	Voie 2	Voie 3	Commande PN
(Ferme ,ouvre) ,	(Ferme ,ouvre)	(Ferme ,ouvre) ,	(fermeture, ouverture)

Dans le cas où les trains circulent dans les deux sens sur les deux voies, il faut ajouter dans CDM Rail sur les voies deux paires d'actionneurs supplémentaires par voie par sens, car les actionneurs sont sensibles au sens de passage du train, et ajouter leurs déclarations dans la ligne. On aura donc 4 actionneurs par voie.

L'ouverture du PN étant provoquée par la retombée du dernier actionneur de sortie (par la queue du train le franchissant), il n'est pas nécessaire de le mettre distant par rapport au PN.

Dans CDM, l'adresse de l'actionneur utilisée est son adresse (ici 809). Laisser cocher « out1 »



Erreurs à la lecture du fichier de configuration

Le fichier de configuration est lu au démarrage du programme signaux complexes. Certaines vérifications de cohérence sont faites lors de la lecture du fichier config.cfg. Une erreur détectée dans celui-ci affichera un message d'erreur.
Les vérifications sont les suivantes :

Les lignes de la section des branches doivent commencer ou se terminer par un aiguillage ou un buttoir.
Un aiguillage n'a pas décrit dans la section des branches du réseau mais il est présent dans la section description des aiguillages.
Un détecteur est décrit dans la description d'un aiguillage mais ce détecteur est absent dans la description des branches.

Erreurs à l'exécution

D'autre part des erreurs peuvent survenir pendant l'exécution du programme. La plupart du temps, elles proviennent d'une erreur de modélisation du réseau dans le fichier de configuration. Il est difficile de créer un fichier config.cfg correct dès le premier essai.
Néanmoins, des erreurs de fond du programme sont la deuxième cause d'erreur, qui ne sont pas liées au fichier. Ces erreurs, beaucoup plus rares, se produisent pour des cas particuliers. Ce sont :

131 - Erreur fatale : adresse nulle sur un aiguillage pris en pointe droit
134 - Erreur fatale : adresse nulle sur un aiguillage pris en pointe dévié
136 - Erreur fatale : adresse nulle sur un aiguillage pris en talon
141 - Erreur fatale : adresse nulle sur un aiguillage bis pris en pointe droit
144 - Erreur fatale : adresse nulle sur un aiguillage bis pris en pointe dévié
146 - Erreur fatale : adresse nulle sur un aiguillage bis pris en talon
139 - Erreur fatale - Aucun cas TJD/S : impossible de résoudre une TJD ou une TJS
Elément n non trouvé : un élément recherché (aiguillage ou détecteur) n'a pas été trouvé dans les branches
Erreur fatale X : Itération trop longue : une recherche sur un élément prend trop de boucles programme
X=300 201 200 201
Erreur X - feu non trouvé
X=600 650
601 - Signal « adresse » non renseigné
980 : Détecteur suivant introuvable - route x à y impossible à déterminer
981 : Détecteur précédent introuvable - route x à y impossible à déterminer

Ces deux erreurs (surtout la 981) surviennent lorsque la position d'un aiguillage ne correspond pas à la réalité à cause du bug de CDM rail qui ne transmet pas la position des aiguillages BIS.

Fichier de configuration client-GL.cfg

Il s'agit d'un deuxième fichier de configuration qui est spécifique à signaux_complexes_GL
Il contient des variables permettant son fonctionnement.

Les variables importantes sont les suivantes :

```
/ Adresse IP V4 du PC sur lequel s'exécute CDM : port
127.0.0.1:9999
```

Il s'agit du PC sur lequel CDM fonctionne. Signaux_complexes_GL utilise un socket (adresse ip + port) pour communiquer avec le serveur de CDM rail. Les deux programmes peuvent donc être sur deux PC différents.
L'adresse 127.0.0.1 représente le PC sur lequel le programme Signaux_complexes_GL s'exécute, lorsque les deux programmes s'exécutent sur le même PC, et c'est donc cette adresse ip qu'il faudra utiliser.

Le port doit être le même que celui de CDM rail (9999 par défaut ; modifiable dans CDM rail dans le menu Comm.IP / Paramétrage de la liaison IP. Signaux_complexes_gl ne gère pas de nom d'hôte.

Uniquement pour les centrales utilisant XpressNet ou compatibles (Lenz / roco):

```
/ Adresse IP V4 de l'interface LI-USB Ethernet : port
/ par défaut le port est 5550
/ ne pas connecter le port ou mettre 0 si on travaille avec l'interface USB
192.168.1.23:5550
```

Si l'on veut utiliser Signaux_complexes_GL de façon autonome, il faut indiquer par quel moyen il va se connecter à l'interface XpressNet : soit par une liaison réseau (wifi ou Ethernet) ou par une liaison USB. Cette ligne contient l'adresse IP et le port de l'interface Lenz LI-USB. L'adresse IP montrée ci-dessus (192.168.1.23) correspond à une interface Lenz_USB-ETH qui a été configurée avec cette adresse évidemment.

Configuration en mode autonome :
Uniquement pour les centrales XpressNet ou compatibles (Lenz / Roco): LI100…Genli

```
/ ========================================================================
/ D é f i n i t i o n  de l'interface XpressNet pour utilisation en mode autonome
/ Adresse IP V4 de l'interface LI-USB Ethernet : port
/ par défaut le port est 5550
/ ne pas connecter le port ou mettre 0 si on travaille avec l'interface USB
192.168.1.23:5550
/
```

Le deuxième moyen d'utiliser Signaux_complexes_GL de façon autonome est d'utiliser le port USB pour qu'il se connecte à l'interface XpressNet. Cette ligne contient le numéro de port USB de l'interface et le protocole à utiliser. Ce numéro de port va de 1 à 9. 0 représente une liaison inutilisée.
La vitesse est à adapter à votre interface :
GENLI : 9600 bauds sans protocole (0)

LI100 : 9600 bauds
LI100F : 9600 ou 19200
LI101 : 19200 ou 38400 ou 57600 ou 115200 bauds protocole matériel rts-cts (2)
La parité est toujours sans (N), 1 bit de stop
Protocole : (0=sans) (1=xon xoff) (2=RTS-CTS) (3=RTS-XonXoff) (4=CTS)

```
/ port COM de l'adresse USB de l'interface LU-USB - LI100 - LI100F - LI101F - GENLI
/ attention de COM1 à 9 - Si le port de l'interface USB >9, il faut le changer
/ manuellement dans le gestionnaire des périphériques
/ mettre 0 si inutilisée
/ Le programme ne tentera pas de se connecter à la centrale si CDM rail est détecté
/ Com:vitesse,parité,nombre de bits,bits de stop,protocole: 0=aucun 1=Xon-Xoff 2=RTS-CTS
3=RTS-Xon-Xoff 4=CTS
/ voir notice page 29
/
COM3:57600,N,8,1,2
```

Il est possible d'indiquer une valeur de temporisation entre deux octets transférés à l'interface. Elle peut être nulle. Pour les interfaces série sans protocole (0) comme le GENLI, il est conseillé de la positionner à une valeur de l'ordre de 30 (ms). Pour les interfaces avec protocole matériel RTS-CTS (2) cette variable est ignorée.

```
/
/ Temporisation en ms d'envoi entre deux octets de la trame
0
```

La variable suivante définit une valeur maximale par tranche de 100 ms qui définit le temps d'attente de la réponse de l'interface après une trame qui lui est transférée. Cette valeur est à tester en fonction de votre interface. En cas de dépassement de la valeur, un message « pas de réponse de l'interface » sera affiché.

```
/ Temporisation maximale de contrôle après non réponse de l'interface, en tranches de 100ms
/ à adapter en fonction de l'interface. Ex 7=700ms d'attente maxi
7
```

La variable entête permet d'intercaler ou non des octets de synchronisation nécessaires aux différentes interfaces. En principe avec l'utilisation d'interfaces série (GENLI), cette valeur doit être à 0. Pour les interfaces utilisant nativement de l'USB, cette valeur doit être à 1. La valeur 2 est utilisée exclusivement pour des interfaces à base d'arduino pour xpressnet.

```
/
/ Entete: préfixe ajouté aux trames :
/ Entete=0 - n'ajoute rien aux trames - pour une interface RS232 ou GENLI
/ Entete=1 - Ajoute FF FE au début de chaque trame envoyée à l'interface  : pour une interface
USB, entete=1
/ Entete=2 - Ajoute E4 au début de chaque trame et 0D 0D 0A en fin de chaque trame : pour
compatibilité arduino
1
```

```
/ Avec (1) ou sans (0) initialisation des aiguillages au démarrage selon le tableau ci après
0
```

En mode autonome, cette variable à 1 permet de lancer ou non (variable à 0) une séquence de positionnement des aiguillages au démarrage selon la description qui suit.

Interfaces XpressNet

Les LI100 (Xbus) LI101F (xpressnet) et LI100F de Lenz sont des interfaces série

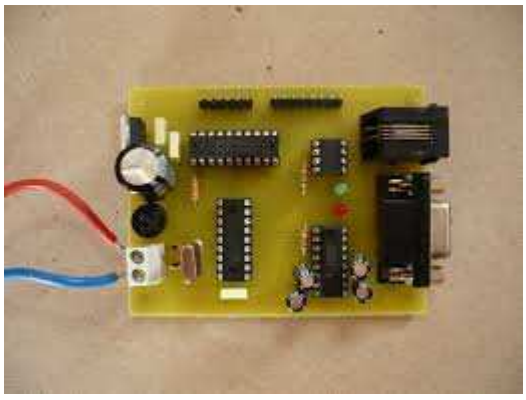
LI100 – LI100F

LI101F

Interface USB (23150)

Interface USB – Ethernet (23151)

L'Interface Genli S88 est à la base une interface série, mais qui peut être dotée d'un convertisseur série-USB (mais elle reste à liaison série à la base)



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.

Utilisation du programme signaux_complexes_GL avec CDM rail

- 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_GL.

Il va alors afficher l'état des connexions :

Exemple 1 :
Erreur 10065 socket IP Lenz: Port non connecté
Erreur 10061 socket IP CDM Rail: Connexion refusée

Le programme signaux_complexes_GL ne s'est pas connecté l'interface LENZ par réseau. Le programme signaux_complexes_GL ne s'est pas connecté à CDM rail (CDM est pas lancé ou son serveur)

Exemple 2 :
Erreur 10065 socket IP Lenz: Port non connecté
CDM Rail connecté avec l'ID XX

Le programme signaux_complexes_GL ne s'est pas connecté l'interface LENZ par réseau. Le programme signaux_complexes_GL s'est connecté à CDM rail

Pour relancer une connexion à CDM rail, il faut le reconnecter en sélection le menu « interfaces / connecter CDM rail »

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

Au lancement du mode TCO ou du RUN, CDM rail positionne les aiguillages. Ces positions sont alors récupérées par le programme client. (ainsi que chaque changement d'aiguillage par la suite).

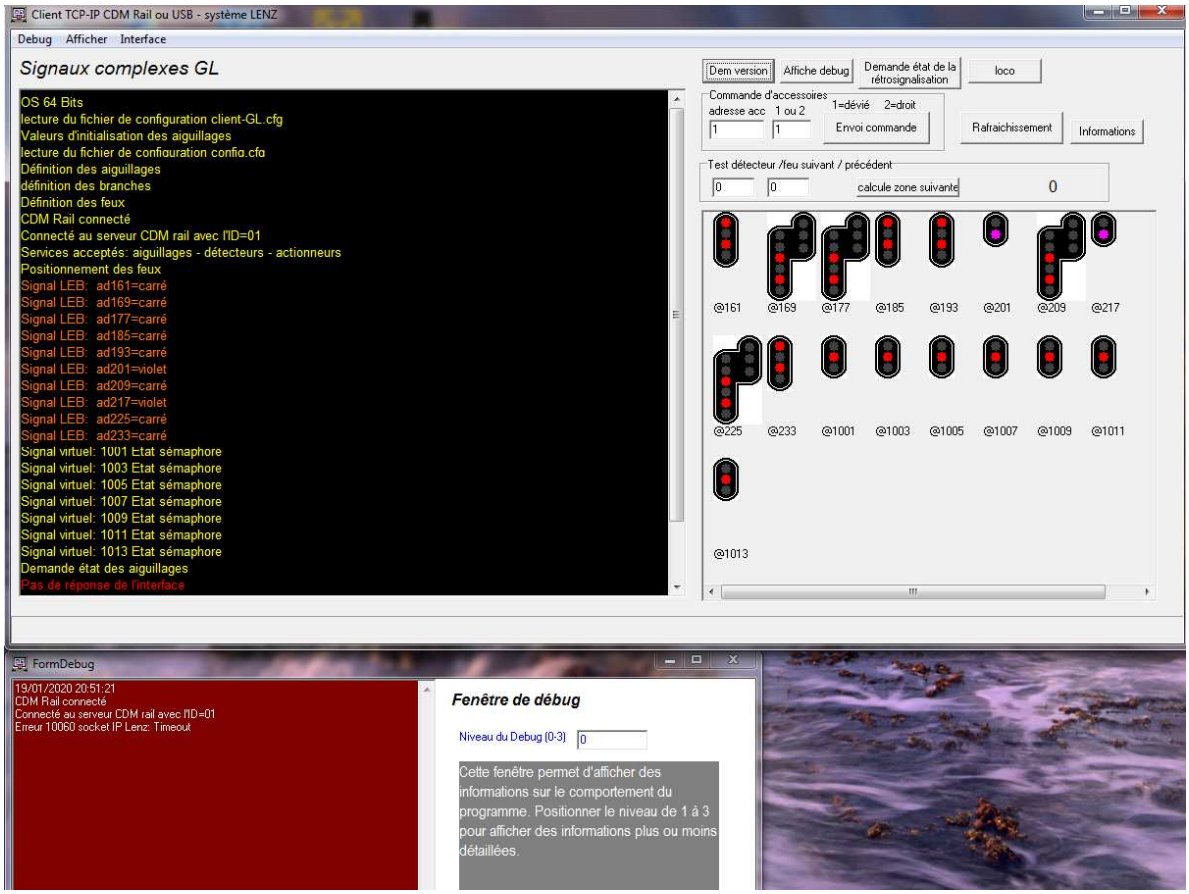
Il faut un passage sur deux détecteurs consécutifs pour synchroniser un train avec le programme.

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 Debug et « afficher changement des détecteurs » (menus à bascule).

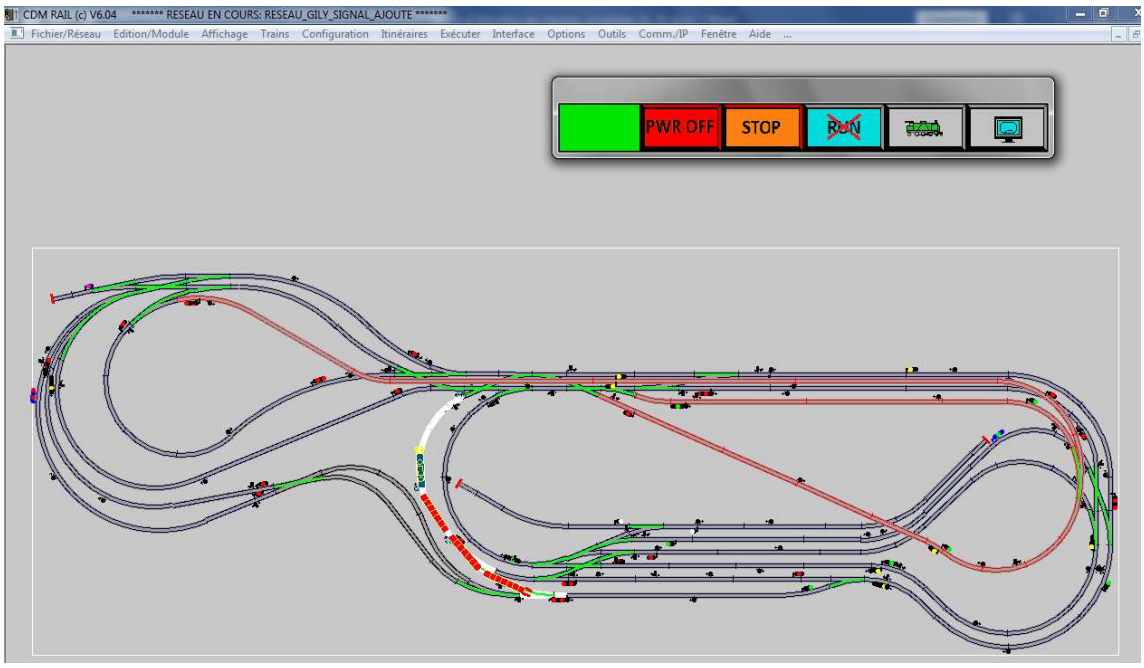
La fenêtre debug peut être fermée à tout instant et rappelée par le bouton « affiche debug »

Utiliser CDM rail et le programme signaux_complexes_GL occupe 3 fenêtres. Il sera donc opportun d'utiliser deux écrans de travail et de répartir les fenêtres sur les deux écrans.

Ecran signaux_complexes_GL



Ecran CDM rail



Pilotage individuel des signaux

En cliquant sur l'image d'un signal dans la partie droite de la fenêtre, on ouvre la fenêtre de pilotage, et on peut sélectionner un aspect pour le signal choisi et envoyer la commande.

Les exclusions sont automatiquement gérées et ne seront pas affichées même si elles sont sélectionnées, comme par exemple ci-dessus : affichage d'un ralentissement 30 avec un avertissement n'est règlementairement pas possible.

Simulateur

Le simulateur permet de simuler des circulations préalablement enregistrées. Dans cette étape, Il n'est pas nécessaire d'être connecté à CDM rail ou à la centrale via XpressNet.

Il faut d'abord procéder à un enregistrement sur le réseau réel sur un parcours :

Faire circuler un ou plusieurs trains (en mode CDM ou sans CDM avec XpressNet). A la fin du parcours, aller dans la fenêtre debug en cliquant sur le bouton « *Affiche debug* » puis cliquer sur le bouton *« Affiche Evts chrono détecteurs* ». Une liste d'évènements détecteurs s'affiche dans la fenêtre de gauche, correspondant au parcours réalisé. Cliquer sur le bouton « *Sauvegarder le log* » Ce bouton sauvegarde le contenu de la fenêtre de gauche ; donner un nom de fichier.

Dans un deuxième temps, à n'importe quel autre moment, on peut resimuler les parcours des trains dans signaux complexes_GL depuis le fichier (les signaux seront pilotés).
Dans la fenêtre principale, sélectionner le menu *Divers / ouvrir un fichier de simulation*. Une fenêtre apparaît. Il est possible de choisir l'intervalle de temps entre deux détecteurs (temps comprimé).

Cliquer sur le bouton « *charger un fichier de simulation* » et sélectionner le fichier précédemment sauvegardé. La simulation commence dans 8 secondes.

Lecture / Ecriture de variables de configuration (CV)

Ne fonctionne qu'avec des centrales dotées de bus **XpressNet version minimale 3.6** ou compatibles.

Il est possible de lire ou d'écrire des variables CV ou un fichier contenant des CV à un décodeur. Pour cela, le décodeur (train, feu) doit être branché sur la voie de programmation via les lignes PQ. Le programme doit être connecté directement à la centrale par USB ou Ethernet. CDM ne doit pas fonctionner (Signaux complexes est en mode autonome). On doit arrêter CDM rail et sélectionner le menu Interface/connecter USB ou interface/Connecter Ethernet pour se connecter à la centrale (pourvu que le fichier de configuration client-GL.cfg soit correctement renseigné : port COM usb ou adresse ethernet de l'interface XpressNet)

L'écriture des CV concerne les adresses de CV de 1 à 255. Pour les adresses >255, il faut utiliser XpressNet V3.6 mais ce n'est pas prévu dans le programme.

Lorsque la centrale est connectée par XpressNet via l'USB ou éthernet au programme (donc sans CDM), le boutons « Ecriture CV par le bus XpressNet » et les menus « lire un fichier de CV vers un accessoire / Lire un accessoire vers un fichier de CV » sont utilisables.

Ecriture d'un CV seul dans un accessoire

Indiquer l'adresse du CV dans le champ Adresse acc et la valeur du CV dans le champ 1ou 2 et cliquer sur le bouton Ecriture CV. La centrale va passer en mode programmation (service mode) et écrire le CV. Pour repasser en mode DCC traditionnel, cliquer sur le bouton « Reprise DCC ».

Ecriture de CV en cascade dans un accessoire

Un catalogue de CV peut être défini pour un accessoire dans un fichier texte en ASCII. Cela évite les manipulations fastidieuses si de nombreux CV doivent être écrits dans un accessoire, et permet de stocker les valeurs dans un fichier pour archivage. Cliquer sur le menu « Lire un fichier de CV vers un accessoire » et choisir le fichier. La centrale va passer en mode programmation (service mode) et écrire les CVs dans l'accessoire branché sur la voie de programmation. Pour repasser en mode DCC traditionnel, cliquer sur le bouton « Reprise DCC ».

La structure du fichier doit contenir 2 champs numériques spécifiant l'adresse du CV suivie de sa valeur. Les champs non numériques sont ignorés, permettant des commentaires. Exemple :

```
Cv   valeur
------aspect1
35    63
36    36
37    0
38    0
------aspect2
39    63
40    33
41    0
```

```
42      0
```

Lecture de CV en cascade depuis un décodeur

Les CV (de 1 à 255) d'un accessoire peuvent lus puis stockés dans un fichier texte en ASCII. Cela évite les manipulations fastidieuses si de nombreux CV doivent être lus depuis un accessoire, et permet de stocker les valeurs dans un fichier pour archivage. Cliquer sur le menu « Lire un accessoire vers un fichier de CV ». La centrale va passer en mode programmation (service mode) et lire les CVs. », choisir le fichier de destination. Pour repasser en mode DCC traditionnel, cliquer sur le bouton « Reprise DCC ».

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 directement CDM rail
3. Fermer le programme client.

Modification du programme avec Delphi 7

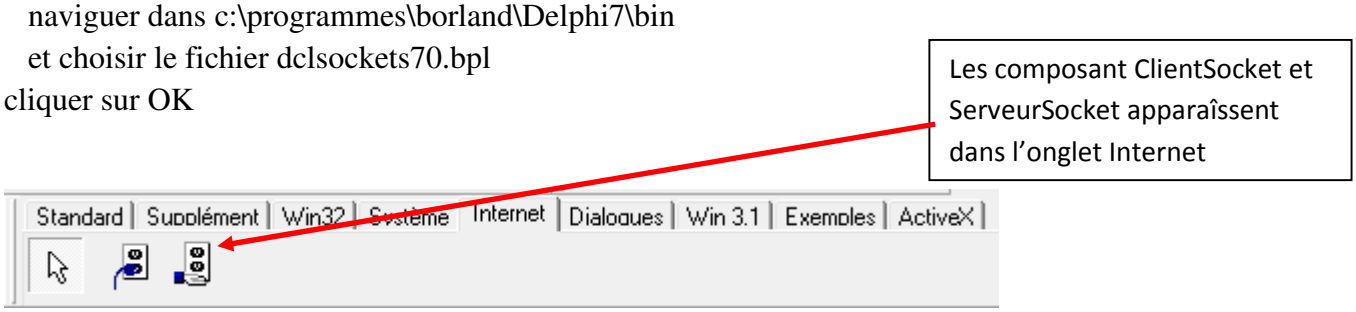
Pour les programmeurs, signaux complexes GL a été écrit en Delphi7. Il existe une notice de programmation pour l'aide à la modification du programme signaux complexes. Delphi7 est téléchargeable gratuitement ici (après avoir créé un compte)
https://delphi.developpez.com/telecharger-gratuit/delphi7-perso/
la clé est la suivante

```
Numéro : YH?Z-2WDEGK-S48529-3AS3
Clé : G5N-D95
```

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

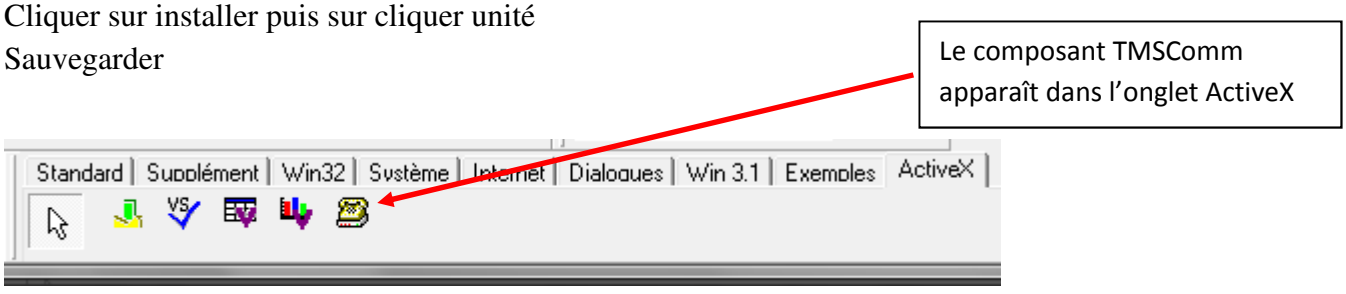
Installation des composants socket (TClientSocket et TServerSocket)

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



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



Debuggeur

Dans outils / options du déboggeur / 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 déja), 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 = {
      2143341208000000ED030000ED03000001568A64000006000000010000040000
      0002000080250000000008000000000000000003F00000011000000}
  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 :