

CDM-RAIL

GESTION DES SIGNAUX

J.P.PILLOU

28 janvier 2016

1 APPROCHE CDM-RAIL POUR LA GESTION DES SIGNAUX

1.1 LA NORME DCC CONCERNANT LES ACCESSOIRES

Le gros problème de la gestion des signaux, en DCC, est qu'il n'existe aucun standard, au-delà de la commande d'accessoire, à deux états.

Cette commande permet de mettre à 1 (ou 0), l'une ou l'autre de deux sorties OUT1 ou OUT2. Les deux commandes ne peuvent pas être à 1 simultanément, de sorte que les états autorisés sont:

OUT1	OUT2
1	0
0	1
0	0

Plus gênant encore, alors que la norme DCC permet de maintenir un état de sortie à 1 sur OUT1 ou OUT2, certaines centrales ne le permettent pas (LENZ en particulier). Autrement dit, dès qu'une sortie d'accessoire est mise à 1, ça remet automatiquement à 0 **toutes les sorties de tous les autres accessoires**. Et par conséquent, ça empêche d'utiliser un codage d'état à 3 valeurs au lieu de 2.

La norme DCC a introduit une commande plus complexe pour piloter des signaux avec un nombre d'états beaucoup plus important, mais cette commande n'est gérée (apparemment) par aucune centrale.... et n'est gérée par aucun décodeur d'accessoire.

Je rappelle qu'un logiciel comme CDM-Rail n'a pas accès directement au DCC. Il est obligé de passer par un protocole de communication compatible avec la centrale: pour Lenz, c'est le protocole Xpressnet. Et si ce protocole de communication ne permet pas d'accéder à certaines commandes DCC, ces commandes sont inaccessibles. A ce titre, le SPROG3 est plus puissant que Lenz, puisque le protocole permet de coder n'importe quelle commande DCC, qui est transmise de façon transparente.

Pour gérer les signaux, on est donc obligé d'envoyer des impulsions sur plusieurs adresses d'accessoires: 2 adresses pour 3 ou 4 états, 3 adresses pour gérer jusqu'à 8 états. Les signaux complexes de la SNCF peuvent dépasser 8 états, ce qui requiert 4 adresses.

Le problème, c'est que chaque fabricant a sa propre convention d'envoi des séquences, sur différentes adresses (heureusement toujours consécutives quand même).

1.2 L'APPROCHE CDM-RAIL

1.3 LES FICHIERS DE TIMING: INTRODUCTION

CDM-Rail, dans sa version actuelle, ne gère que trois états, qui sont les états du système BAL. Mais on imagine assez facilement étendre cette gestion à l'état "carré".

L'approche consiste à passer par des fichiers de description de timing (de commandes), à envoyer sur la ou les adresses d'un décodeur d'accessoire

Il existe, à ce jour, 5 fichiers correspondant à cinq type d'accessoires différents. Ces fichiers se trouvent dans le sous répertoire "Data" du répertoire d'installation de CDM-Rail.

signal_timing_00_2_Standard.txt:	décodeur "standard" (LDT SA-DEC-4)
signal_timing_01_2_LDT_LS_DEC.txt:	décodeur de signaux LDT LS_DEC
signal_timing_02_2_CDF.txt:	décodeur de signaux CDF
signal_timing_08_1_CDM1.txt:	décodeur 3 états 1 seule adresse (approche perso).
signal_timing_09_1_CDM2.txt:	décodeur 3 états 1 seule adresse (approche chris68).

Le nommage de ces fichiers est de la forme

signal_timing_<numéro timing>_<nombre d'adresses>_<nom du type de timing>.txt

- Le numéro de timing est écrit sur deux chiffres (en décimal), et va de 00 à 15. Il y a donc jusqu'à 16 fichiers de timings différents possibles.
- le nombre d'adresses (sur un seul chiffre) est le nombre d'adresses consécutives utilisées pour attaquer le décodeur.
- Enfin, la chaîne de caractères qui termine le nom du fichier (avant le point) est celle qui apparaît dans la fenêtre déroulante de liste des types de timings, visible en configuration de signal.

1.4 FICHIERS DE TIMING: CONTENU

1.4.1 Exemple N°1: fichier signal_timing_00_2_Standard.txt

A titre d'exemple, voici le contenu du fichier **signal_timing_00_2_Standard.txt**

DO NOT MODIFY!!! NE PAS MODIFIER !!

```
4
0      0      1      0      1
0      0      0      1      0
1      0      1      0      1
1      0      0      1      0
4
0      1      1      0      1
0      1      0      1      0
1      0      1      0      1
1      0      0      1      0
4
0      0      1      0      1
0      0      0      1      0
1      1      1      0      1
1      1      0      1      0
```

La ligne commençant par # est un commentaire

Puis le format est le suivant:

<nombre de commandes pour l'état STOP (4 dans ce cas)
<1ere commande STOP>

...

<4eme commande STOP>

<nombre de commandes pour l'état GO (4 dans ce cas)
<1ere commande GO>

...

<4eme commande GO>

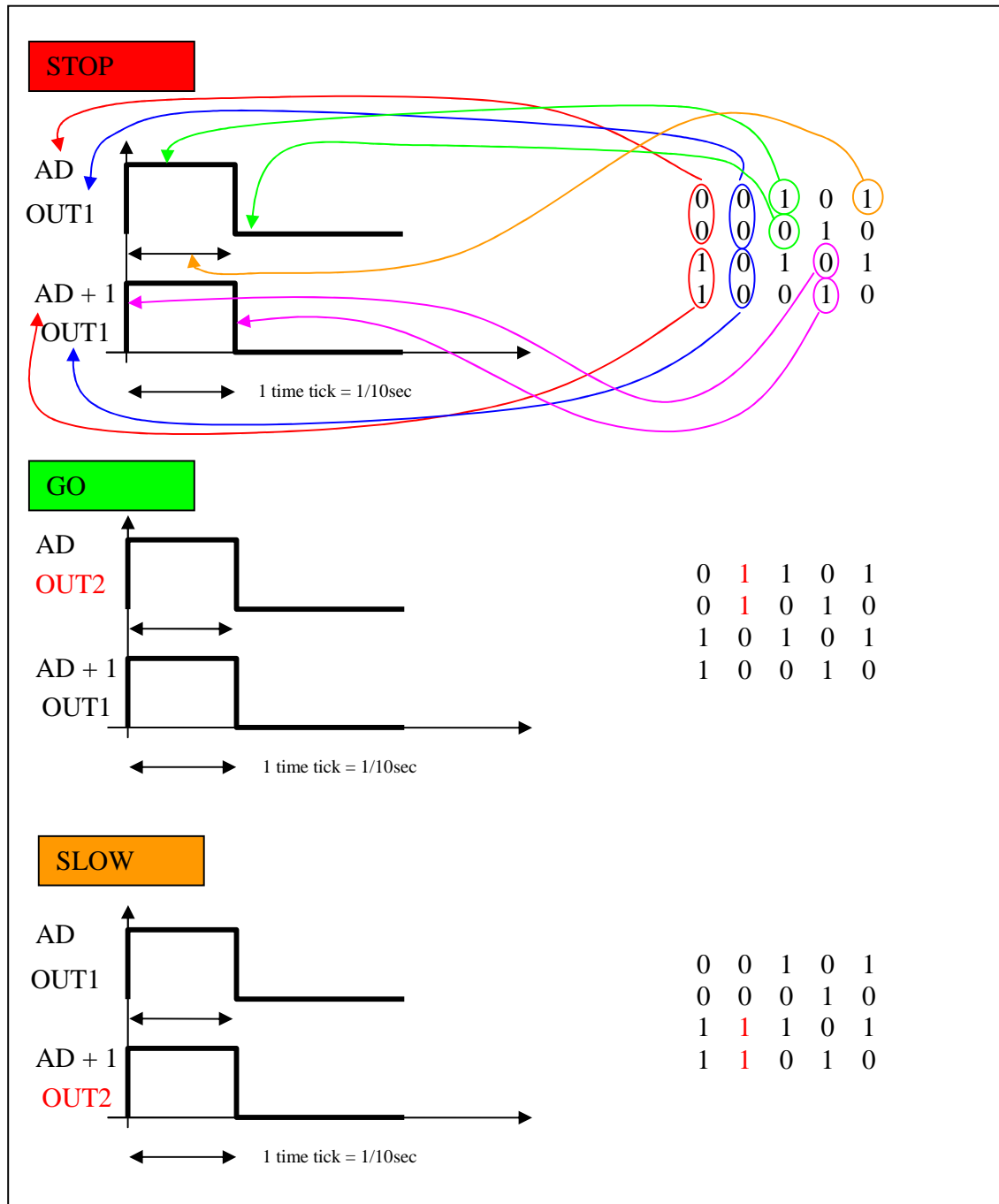
<nombre de commandes pour l'état SLOW (4 dans ce cas)
<1ere commande SLOW>

...

<4eme commande SLOW>

Et le format de chaque ligne de commande est

<offset d'adresse> <OUT2 (1) ou OUT1 (0)> <actif (1) ou non (0)>
 <décalage en timeticks (1/10s) par rapport au début de la séquence>
 <durée de l'impulsion en timeticks>



Note 1: L'information de durée d'impulsion est redondante avec l'information de temps de retard du front de redescende. En réalité, elle n'a été introduite que pour différencier une impulsion (auquel cas sa valeur est "1"), d'un état continu (valeur "0").

Note 2: ce format permettrait de décrire des séquences plus longues. Par exemple l'adresse AD pourrait décrire une horloge, et l'adresse AD+1 une donnée, pour envoyer un code binaire sur N bit en série. Il faudrait seulement modifier la dimension du tableau de stockage interne dans CDM-Rail.

1.4.2 Exemple N°2: fichier signal timing_08_1_CDM1.txt

Le contenu du fichier signal_timing_08_1_CDM1.txt est le suivant (à partir de la V5.32c)

Comme l'indique le champs de nombre d'adresses, ce format est fait pour gérer 3 états avec une adresse unique.

DO NOT MODIFY!!! NE PAS MODIFIER !!

```
1
0      0      1      0      0
1
0      1      1      0      0
2
0      0      0      0      0
0      1      0      0      0
```

- L'offset d'adresse est toujours 0 (adresse unique)
- Le dernier paramètre (durée d'impulsion) est toujours à 0, indiquant un état continu, et non une impulsion.
- La première section (STOP) comporte une commande unique de mise à 1 de la sortie OUT1.
- La deuxième section (GO) comporte une commande unique de mise à 1 de la sortie OUT2.
- La troisième section (SLOW) comporte deux instructions de mise à 0 des sorties OUT1 et OUT2.

Rappelons que ce type de commandes n'est pas compatible avec les centrales évoluées (Lenz,...) qui considèrent qu'une seule sortie de décodeurs d'accessoires (même à des adresses différentes), peut être à l'état 1.

Mais ça peut permettre des décodeurs spécifiques simples et efficaces, lorsqu'on utilise (paradoxalement) des centrales bas de gamme (Multimaus, SPROG3), qui permettent de

maintenir un état 1 sur une sortie "accessoire", et d'envoyer explicitement une commande de mise à 0 sur le DCC (ce que ne fait pas la centrale Lenz).

1.4.3 Exemple N°3: fichier signal timing 09 1 CDM2.txt

Il s'agit ici d'un protocole proposé par Chris68, pour permettre de coder 3 états en n'utilisant qu'une seule adresse, tout en restant compatible avec les centrales telles que Lenz.

DO NOT MODIFY!!! NE PAS MODIFIER !!

2

0 0 1 0 1

0 0 0 1 0

2

0 1 1 0 1

0 1 0 1 0

3

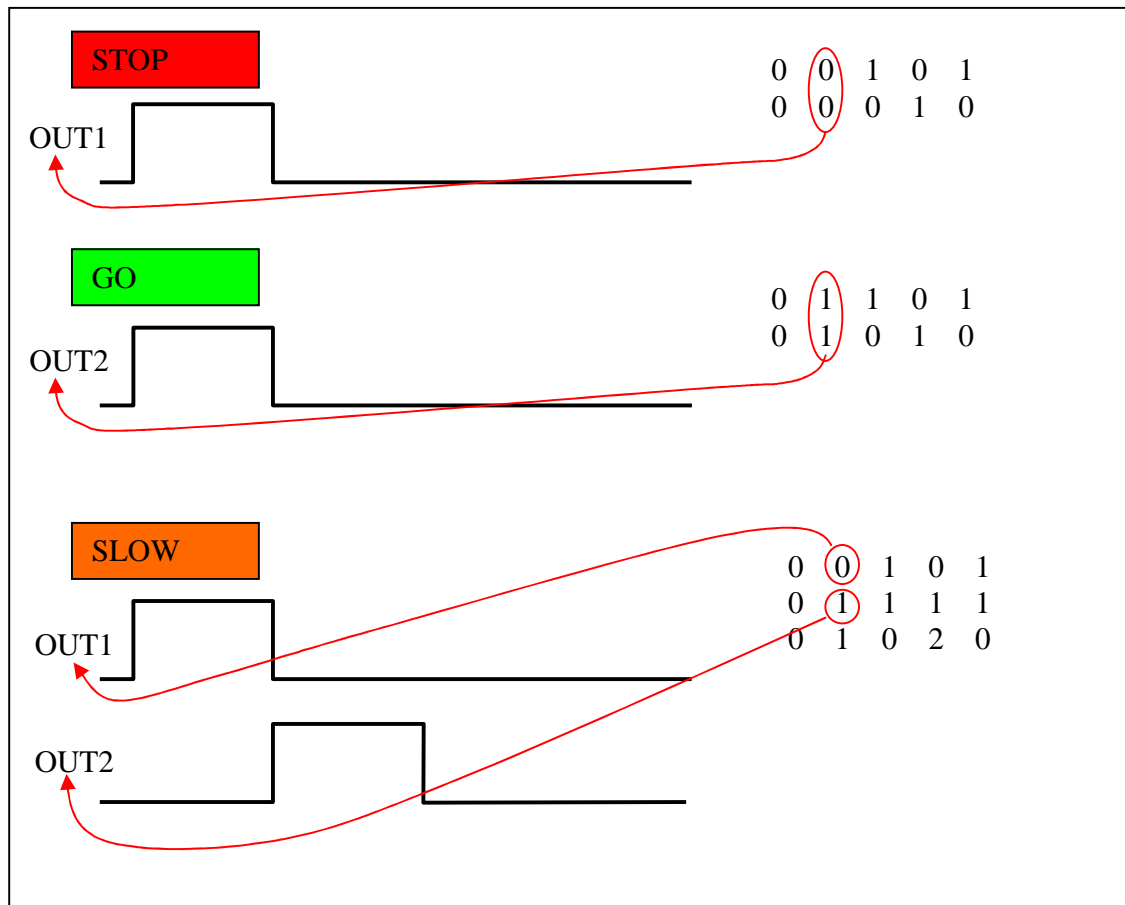
0 0 1 0 1

0 1 1 1 1

0 1 0 2 0

- Etat STOP: on envoie une impulsion sur la sortie OUT1 de l'adresse unique.
- Etat GO: on envoie une impulsion sur la sortie OUT2 de l'adresse unique.
- Et pour l'état SLOW, on envoie successivement une impulsion sur OUT1 immédiatement suivie d'une impulsion sur OUT2.

A charge au décodeur de mesurer l'intervalle de temps entre les deux impulsions pour différencier les deux impulsions de l'état SLOW, de deux impulsions isolées correspondant au STOP et au GO



1.5 FICHIERS DE TIMING "USER"

Cette approche permet à tout utilisateur qui voudrait créer son propre décodeur d'accessoires, de faire générer les commandes spécifiques à ce décodeur.

Utiliser de préférence les derniers numéros de fichiers de timing: 14 et 15.
En tout cas, des numéros non encore utilisés.