MARKLIN : PROTOCOLE DE COMMUNICATION CAN et ETHERNET

Préambule :

Ce document est une traduction du document Marklin intitulé « cs2can-protokoll-2_0 » que j'ai commencé à compléter avec des commentaires, quelques explications et du code c++ pour donner quelques exemples de mise en pratique.

La traduction peut présenter des erreurs mais reste cohérente. Les ajouts permettront de comprendre le sens exact des choses.

Le document original est disponible ici dans sa version originale :
https://streaming.maerklin.de/public-media/cs2/cs2CAN-Protokoll-2_0.pdf

Dans ce qui suit, les informations directement traduites du document Marklin sont en *italique*. Les numérotations de sections et les titres ont été conservés pour faciliter la mise en relation des deux documents et respecter la cohérence de la structure originale.

Il est quelques fois fait référence à la bibliothèque **Railuino** de Joerg Pleumann car c'est le travail de référence pour ce qui est de piloter en **D**o **I**t **Y**ourself des locomotives au travers du bus CAN de Marklin. C'est une bibliothèque écrite en C++ et utilise des classes (et objets donc) qui est selon moi très bien écrite. Mais elle a maintenant 11 ans d'âge et n'incorpore pas, en particulier, les nombreuses avancées des normes C++11 et suivantes. Sont également apparus de nouveaux matériels comme l'ESP32 (avec lequel Railuino n'est pas compatible). J'ai donc entrepris une réécriture assez fondamentale qui est publiée ici : https://github.com/BOBILLEChristophe/Railuino

J'ai aussi découvert au cours de mes recherches l'énorme travail réalisé par Gérald Litzistorf https://gelit.ch qui reprend et prolonge le travail de Joerg Pleumann jusqu'à maitriser non seulement la communication CAN mais aussi la communication de voie ce qui vraisemblablement ouvre des perspectives très intéressantes pour le pilotage et également l'animation des réseaux. Mais cela dépasse le cadre de ce document qui est centré sur la communication CAN.

Cette traduction ne porte que sur environ la moitié de la documentation d'origine, en fait les parties dont j'avais besoin. Je poursuivrai sans doute, probablement plus pour illustre le document de base un peu ardu, par des exemples de code c++ et des commentaires.

La date en bas de page est mise à jour pour chaque édition. C'est elle qui vous permettra de savoir si vous disposez de la version le plus récente.

Table des matières :

Clause de non-responsabilité :

Les informations contenues dans ce document décrivent la systématique de communication des composants de Märklin Digital. Cette documentation est fournie "telle quelle" sans aucune garantie quant à son fonctionnement, à sa justesse ou à son absence d'erreurs. L'auteur ne peut être tenu responsable de tout dommage direct ou indirect - en particulier des dommages à d'autres logiciels, des dommages matériels, des pertes d'utilisation et des dysfonctionnements des produits associés.

Cette description a été développée avec le plus grand soin, cependant, des erreurs ne peuvent jamais être exclues. Par conséquent, aucune garantie ne peut être donnée quant à l'exactitude des informations.

Cette description ne constitue pas une garantie des propriétés du système Märklin Digital ou des produits associés. Des modifications techniques sont réservées.

Droit d'auteur

1 - Connexion CAN

Le bus CAN sert de réseau de communication entre les appareils de commande de Märklin Systems. L'objectif est de mettre à disposition de tous les appareils de commande d'un réseau ferroviaire miniature un moyen de communication uniforme.

Les tâches de commande sont transmises au moyen de messages CAN.

Les mises à jour et les données de configuration sont transmises au moyen de flux CAN. Le débit de données est de 250 kbit/s, la longueur maximale du bus est de 100m.

1.1 - CAN Format de base

Le protocole CAN prescrit que les messages se composent d'un identifiant de message de 29 bits, d'une longueur de message de 4 bits et de 8 octets de données au maximum.

L'identifiant du message est divisé en sous-domaines : priorité (Prio), commande (Command), réponse (Response) et hachage (Hash). La communication est basée sur le format de données suivant :

Identifiant du message 29 bits				Longueur Data	Data							
Priorité	Commande	Reponse	Hash	Data Length Code (DLC)	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Priorité du message	Commande	Commande (0) ou réponse (1)	Résolution des collisions	Nombre de données en bytes 0 à 8	Données	...						

1.2 - Description de base des champs

1.2.1 Priorité (Prio)

Détermine la priorité du message sur le bus CAN :

Dans une application temps réel, certaines informations nécessitent d'avoir la priorité et la garantie d'être délivrées avec une sécurité maximale. Dans la norme CAN, les messages disposant des identifiants les plus faibles sont prioritaires par rapport à ceux disposants d'identifiants plus « forts ».

0x00…… est prioritaire par rapport à 0x01…, lui-même prioritaire par rapport à 0X02…

Prio 1 :	Arrêt / Go / Message de court- circuit
Prio 2 :	Réactions
Prio 3 :	Arrêter la locomotive
Prio 4 :	Locomotive / commandes d'accessoires

1.2.2 - Commande

Détermine la commande à exécuter par le terminal ou la commande exécutée.
Les valeurs de commande sont clairement définies.
Plages de valeurs caractéristiques pour les valeurs de commande :

Domaine	Nombre	Début	Fin
Commandes système	1	0x00	0x00
Administration	8	0x01	0x0A
Accessoires	2	0x0B	0x0D
Réactions	-	0x10	0x12
Mise à jour du logiciel / Config	6	0x18	0x1C
Transmission d'informations par GUI	3	0x20	0x22
Commandes d'automatisation		0x30	

1.2.3 - Réponse

Détermine si le message CAN est une demande ou une réponse à une demande précédente. En principe, une demande est lancée sans que le bit de réponse ne soit activé. Dès qu'une commande a été exécutée, elle est confirmée avec le bit de réponse activé, ainsi que le contenu initial du message ou les valeurs demandées. Chaque participant au bus qui a exécuté le message confirme une commande.

Voici par exemple deux messages de ping.

Le premier message est une demande :

```
Frame ID        : 0x302F39
Priorité        : 0x00
CAN_Command     : 0x18
Hash            : 0x2F39          // ESP32
Frame length    : 0               // Pas de data
Response        : 0               //Bit réponse à 0, c'est une commande
Data :          : Aucune data
```

*Voici l'identifiant de ce message en binaire : 11000*0*0010111100111001*
Le bit 17 repéré en rouge est à 0.
Il n'y a par ailleurs aucune data ce qui est caractéristique d'une demande.

En C++ la méthode setPing() pourrait donc s'écrire :

```
bool setPing()
{
   TrackMessage message;
   message.command = 0x18;
   message.length = 0;
   message.response = false;
   // Pas de data
   ctrl.sendMessage(message);
   return true;
}
```

Le second message est la réponse en provenance d'un appareil relié au bus CAN, il s'agit ici de la MS2 :

Fram ID : 0x318B23
Priorité : 0x00
CAN_Command : 0x18
Hash : 0x3363 // MS2 60657
Frame length : 8 // 8 octets (bytes) de données
Response : 1
Data : 0x4D 0x56 0x1C 0x75 0x03 0x94 0x00 0x33

*Voici l'identifiant de ce message en binaire : 11000**1**1000101100100011*
Le bit 17 repéré en rouge est à 1.

En C++ la commande decodePacket (TrackMessage &inMessage) qui analyse les messages reçus va donc chercher à vérifier :
1° Qu'il s'agit bien de la même commande : inMessage.command == 0x18
2° Que le tableau des datas contient bien 8 éléments
3° Qu'il s'agit bien d'une réponse : inMessage.response == true, information qui sera obtenue par la lecture du bit 17 de l'identifiant :
inMessage.response = inMessage.id & 0x10000 ;

Si vous êtes amené à préciser par programmation qu'il s'agit d'une réponse, il vous faudrait alors réaliser un ***set bit*** *sur le frame.id soit :* ***frame.id |= (1 << 17);***

Le code suivant affiche dans le moniteur série les données du message.

```
void decodePacket(TrackMessage &inMessage)
{

  if( (inMessage.command == 0x18) && (inMessage.length == 8) && (inMessage.response == true))
  {
    /* mfx discovery */
    Serial.print("@PING,");
    Serial.print(inMessage.data[0], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[1], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[2], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[3], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[4], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[5], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[6], HEX);
    Serial.print(",");
    Serial.print(inMessage.data[7], HEX);
    Serial.println(",");

  }
}
```

Le bit contenant la réponse peut être lu avec le code suivant :

```
bool response = (inMessage.id & 0x10000) >> 16;
Serial.printf("Response %d\n", response);
```

Pour chaque commande et pour chaque réponse, Marklin fournit un tableau pour interpréter les données. Pour la réponse à cette commande 0x18, on voit par exemple que les quatre premiers bytes de la réponse nous fournissent l'UID de l'appareil, les bytes 4 et 5, le numéro de version du logiciel et les 2 derniers bytes, l'identifiant de l'appareil qui est en réalité le type de l'appareil, une boîte de voies 60112 et 60113, une MS2 etc…

Identifiant du message 29 bits				Longueur Data	Data							
Priorité 2+2 bits	Commande 8 bits	Response 1 bit	Hash 16 bits	Data Length Code (DLC) 4 bits	Data 0 8 bits	Data 1 8 bits	Data 2 8 bits	Data 3 8 bits	Data 4 8 bits	Data 5 8 bits	Data 6 8 bits	Data 7 8 bits
Priorité du message	Commande	Commande (0) ou réponse (1)	Résolution des collisions	Nombre de données en bytes 0 à 8	Données	…						
0x318B23				8	UID de l'appareil expéditeur				Numéro de version du logiciel		ID de l'appareil	
	0x18		0x3363		0x4D561C75				0x0394		0x0033	
0b0000	0b00011000	0b1	0b1000101100100010									

1.2.4 – Hash

Le hachage remplit une double fonction :

Il sert principalement à résoudre les collisions entre les messages et à garantir l'absence de collision avec le protocole CS1.

Secondairement, il peut contenir le numéro de séquence d'une transmission de données.

Absence de collision avec le protocole CS1 : Dans le protocole CAN du CS1, la valeur 6 n'est pas utilisée pour la "zone com de l'ID", il s'agit des bits 7..9, c'est-à-dire le bit le plus élevé de l'octet le plus bas (0b0xxxxxxx) et les deux bits au-dessus (0bxxxxxx11).

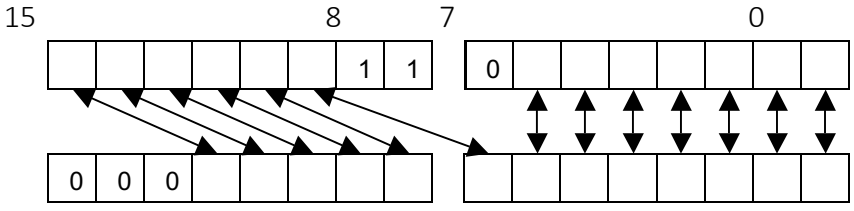
Cette combinaison de bits est donc utilisée de manière fixe dans le hachage pour les différencier

Résolution des collisions : Le hachage sert à rendre les messages CAN exempts de collision avec une probabilité élevée. Cette valeur de 16 bits est formée à partir du hachage de l'UID.

Calcul : 16 bits High UID XOR 16 bits Low de l'UID. Ensuite, les bits sont définis en fonction de la distinction CS1.

Chaque participant au bus doit vérifier l'absence de collision dans le hachage des messages CAN reçus. Si le propre hachage est reçu, il faut en choisir un nouveau. Celui-ci ne doit correspondre à aucun autre reçu.

Numéro de séquence d'une transmission de données : Si le hachage est utilisé pour identifier le numéro de paquet, ces bits sont masqués lors du calcul du numéro de paquet. C'est-à-dire que pour le nombre de 16 bits, les bits 7 à 9 sont masqués, les 3 bits du haut sont à
0. La plage de valeurs se réduit en conséquence à 8192.



Les appareils en DIY que nous ajoutons comme l'ESP32 n'ont pas d'UID « Marklin ». Cependant, Marklin a réservé une plage d'adresse UID pour les particuliers et les clubs (voir 1.3.1.2 ci-dessous). A vous de gérer

vos propres UID.

Dans Railuino, la procédure de détermination du hash se déroule donc en deux temps :

1° - Un hash « arbitraire » sur 16 bits peut être renseigné lors de la création d'une instance de la classe TrackController. L'instanciation de l'objet peut aussi se faire sans renseigner ce hash qui sera alors calculé dans la méthode de classe begin() appelant elle-même la méthode generateHash().

2° - La méthode de classe generateHash() se charge de créer et de vérifier sur le bus si ce hash n'est pas déjà attribué à un autre matériel et, dans l'affirmative, procédera alors à une nouvelle génération.

Ces différentes méthodes seront vues en détail plus tard.

1.2.5 Confirmation

L'initiateur d'un message doit veiller à ce que l'action souhaitée soit effectivement exécutée. Les messages ne sont pas transmis de manière sécurisée via le bus CAN. La réception d'un message n'est pas confirmée. L'exécution d'une commande est confirmée ou acquittée uniquement par l'envoi du message de confirmation. Si cet acquittement fait défaut, on peut supposer que l'action n'a pas été exécutée.

Ce qu'il faut comprendre c'est qu'en CAN, il n'existe aucun mécanisme de confirmation de réception d'un message. Cela est inhérent à la conception même du CAN qui est une messagerie « broadcast ».

Dans le système Marklin, l'envoi d'une commande, si elle est bien reçue par son destinataire et normalement bien exécutée, est systématiquement confirmée. Il appartient donc à l'émetteur (ou un autre mécanisme) de surveiller la réception du message de retour et de prendre les dispositions en conséquence. Par exemple un nouvel envoi de la commande en cas de non-réponse dans un délais donné.

Voici par exemple le tracking de l'envoi d'une commande pour l'activation de la fonction 3 pour la locomotive dont le local ID est 0x4005 :

```
17:49:06.694 -> Frame ID : 0xC8B23
17:49:06.694 -> CAN_Command = 0x06
17:49:06.694 -> Hash = 0x8B23
17:49:06.694 -> Frame len 6
17:49:06.694 -> Response 0
17:49:06.694 -> Data : 0x00 0x00 0x40 0x05 0x03 0x01
```

Et la réponse :

```
17:49:06.694 -> Frame ID : 0xD3363
17:49:06.694 -> CAN_Command = 0x06
17:49:06.809 -> Hash = 0x3363
17:49:06.809 -> Frame len 6
17:49:06.809 -> Response 1
17:49:06.809 -> Data : 0x00 0x00 0x40 0x05 0x03 0x01
```

Les deux messages sont similaires sauf le hash (et l'identifiant qui inclus le hash !) et le champ « Response » qui est à 0 dans le premier message et à 1 dans le second.

Ils émanent donc bien d'un premier appareil, la MS2 pour l'envoi de la commande et de la Gleisbox pour le second message dont le champ « Response » a bien 1 pour valeur.

C'est la Gleisbox qui répond. Marklin spécifie bien que les locomotives ne répondent jamais en direct. Mais l'adresse du décodeur est dans les datas de l'envoi et de la réponse.

Railuino dispose d'un mécanisme de contrôle de l'exécution des commandes au travers de la méthode exchangeMessage() de la classe TrackController qui sera également détaillé plus tard.

1.2.6 - Autres

- Aucune adresse d'expéditeur + de destinataire n'est utilisée dans la communication.

C'est une règle du protocole CAN. Marklin démontre bien sa volonté de respecter la norme. Pour la raison donnée plus précédement, plusieurs appareils peuvent être intéressés par un même message. Il n'y a jamais de destinataire « nommé » pour un message. C'est à chaque appareil de s'assurer si le message envoyé l'intéresse ou pas.

- Aucune trame distante (= demander l'ID CAN au lieu d'envoyer des données avec) n'est utilisée dans la communication. En général, les participants sont configurés de manière à ne pas les recevoir.
- L'ordre des octets dans les messages est toujours Motorola Big Endian.

Cela veut dire que l'octet de poids fort est contenu dans l'indice de data le plus faible. Soit par exemple le nombre 733295205870 qui en HEX s'écrit 0xAABBCCDDEE sera représenté comme ceci :

Data[0] = 0xAA ;

Data[1] = 0xBB ;

Data[2] = 0xCC ;

Data[3] = 0xDD ;

Data[4] = 0xEE ;

1.2.7 - Transmission des commandes CAN via Ethernet

La passerelle Can-UDP peut être démarrée sur le CS2 - via le setup / IP - paramètres. On peut y spécifier une adresse IP (également broadcast) à laquelle la passerelle envoie. Les adresses de port ne sont pas réglables via l'interface et sont fixées sur les ports 15731 et 15730.

Fonctionnement :

Une fois lancée, la passerelle écoute sur le port de réception Ethernet 15731. Elle rejette tous les paquets UDP dont la longueur est différente de 13. Les paquets de longueur 13 sont interprétés comme des paquets Can-Bus : 4 octets d'ID Can-Bus (BigEndian ou Network-Order), 1 octet de longueur et 8 octets de données, à compléter éventuellement par des octets zéro. Ce paquet est ensuite envoyé sur le Can-Bus en tant que message Can-Bus. Les bits ou octets à ne pas représenter sur le bus CAN ne sont pas pris en compte et doivent être mis à "0".

Inversement, la passerelle lit tous les messages Can-Bus, les convertit de manière analogue en paquets UDP de longueur 13 et les envoie à l'adresse IP et au port d'envoi spécifiés (15730).

Exemple de configuration dans le réseau local avec le segment de réseau 192.168.2.0
CS2 : (192.168.2.20) reçoit sur le port 15731, envoie à l'adresse de diffusion 192.168.2.255:15730. PC1 : (192.168.2.10) reçoit sur le port 15730 envoie à CS2 (192.168.2.20:15731)

PC2 : (192.168.2.11) reçoit sur le port 15730 envoie à CS2 (192.168.2.20:15731)

Dans Ethernet, les paquets de 13 octets sont toujours transmis, indépendamment de la taille du datagramme CAN, car la passerelle CAN Ethernet rejette les paquets d'une autre longueur.

Les octets du message CAN sont emballés dans le paquet UDP de la manière suivante :

- Les octets 1 à 4 constituent l'identifiant du message.
- L'octet 5 correspond au DLC du message CAN.
- Les octets 6 à 13 sont les données utiles correspondantes. Les octets non utilisés doivent être remplis par 00.

1.3 - Généralités

1.3.1 - Adresses dans le système - L'UID

L'espace d'adressage total est de 2**32 adresses (0x0000 0000 - 0xFFFF FFFF), ce qui représente environ 4 milliards d'adresses.

Le principe d'un UID est d'être absolument unique. Chaque appareil, chaque décodeur possède donc ainsi un numéro qui le distingue de tous les autres (comme une plaque minéralogique sur une voiture). Que vous ayez deux décodeurs identiques, fabriqués le même jour, à la même heure, ils auront un UID distinct.

1.3.1.1 - Définition des identifiants des participants

Dans le système, chaque participant adressable possède une adresse unique de 32 bits. On distingue les IDE suivants :

UIDE de l'appareil ID universel attribué de manière univoque.

Loc-ID : (= Local ID, pas Locomotive ID) ID local calculé à partir du protocole et de l'adresse.

MFX-UID : MFX Universal ID, identifiant unique d'un participant MFX.

Certains UID d'appareils ont une signification particulière :

L'UID 0x00000000 est l'adresse de diffusion. Il signale que plusieurs participants doivent traiter la même commande.

L'UID 0xFFFFFF n'est pas valable et représente un UID non initialisé du terminal.

1.3.1.2 – Intégration des protocoles de voies existants, formation du "Loc-ID"

L'espace d'adressage compte environ 4 milliards d'adresses disponibles. Une partie de cet espace d'adressage (adresse 0 - 65536) est utilisée pour l'intégration des protocoles existants : Les protocoles numériques existants sont intégrés dans cette zone réservée, représentée par le Loc-ID. Le protocole est déterminé par sa position. Les deux octets inférieurs du Loc-ID sont indiqués pour ces protocoles, les octets supérieurs sont = 0x0000.

On obtient ainsi le schéma d'adresses suivant :

Adresse départ	Adresse finale	Protocole
0x0000	0x03FF	MM1,2 locomotives et décodeurs de fonctions (20 & 40 kHz, 80 & 255 adresses)
0x0400	0x07FF	*Réservé*
0x0800	0x0BFF	SX1
0x0C00	0x0FFF	*Réservé*
0x1000	0x13FF	*Rés. pour MM1,2 Décodeur de fonctions F1 - F4 (40 kHz, 80 & 255 adresses)*
0x1400	0x17FF	*Réservé*
0x1800	0x1BFF	Libre pour les particuliers / clubs
0x1C00	0x1FFF	Libre pour les entreprises
0x2000	0x23FF	*Réservé aux décodeurs de locomotive MM1,2 (20 kHz, 80 & 256 adresses)*
0x2400	0x27FF	*Réservé*
0x2800	0x2BFF	SX1 - Accessoire (extension)
0x2C00	0x2FFF	*Réservé aux tractions (identifiants GUI internes)*
0x3000	0x33FF	MM1,2 Décodeur d'article accessoire (40 kHz, 320 & 1024 adresses)
0x3400	0x37FF	*Réservé*
0x3800	0x3BFF	Accessoires DCC
0x3C00	0x3FFF	Accessoires DCC
0x4000	0x7FFF	MFX
0x8000	0xBFFF	SX2
0xC000	0xFFFF	DCC

Example (Hex) :

Märklin Motorola avec adresse 2 : Base : 00 00 00 Plus adresse : 00 00 00 02

MM2 Accessoires avec adresse 3 : Base : 00 00 30 00 Plus adresse : 00 00 30 03

Dans l'exemple présenté en début de document, nous avions en data[2] la valeur 0x40 qui représente bien l'adresse de départ en MFX plus en data[3], 0x05 qui est l'adresse Loc-ID

```
17:49:06.694 -> Frame ID : 0xD3363
17:49:06.694 -> CAN_Command = 0x06
17:49:06.809 -> Hash = 0x3363
17:49:06.809 -> Frame len 6
17:49:06.809 -> Response 1
17:49:06.809 -> Data : 0x00 0x00 0x40 0x05 0x03 0x01
```

Notez qu'une plage de Loc-ID de 0x1800 à 0x1BFF est réservée pour les particuliers et les clubs. C'est donc dans cette plage que vous pourrez puiser vos adresses pour les appareils DIY.

1.3.2 - Création de l'UID automatique et de l'UID de confirmation.

Dans le système global, il peut y avoir plusieurs appareils ayant la capacité d'une automatisation ou d'un retour d'information.

Pour pouvoir utiliser ces ressources dans l'ensemble du système, cette capacité doit pouvoir être adressée. Pour ce faire, la possibilité de confirmation et d'automatisation est regroupée dans un espace d'adressage commun.

Ces identifications se composent de deux identifications partielles de 16 bits : Un identifiant d'appareil attribué et un identifiant pour l'identification automatique/de retour dans cet appareil. Il est ainsi possible d'interconnecter 65K appareils avec 65K fonctions chacun.

L'adresse de contact ainsi que l'adresse d'automatisation sont donc formées par une combinaison de 16 bits d'identifiant d'appareil et de 16 bits d'identifiant de contact / d'automatisme. Cet adressage permet d'accéder à tous les contacts et fonctions d'automatisation du système.

Grâce à ces identifiants, il est possible d'utiliser les ressources d'un appareil pour les contrôler dans un autre appareil.

1.3.2.1 - Identification des appareils

Les ressources des systèmes qui possèdent un bus S88 / bus de retour d'information ou qui réalisent une fonction d'automatisation se voient attribuer un identifiant d'appareil de 16 bits.

La centrale maître attribue l'identifiant aux terminaux au démarrage du système. Il n'y a pas d'enregistrement résistant à la réinitialisation dans les appareils.

Cette gestion centrale des identifiants des appareils permet de remplacer un appareil défaillant dans le système. L'adressage enregistré et utilisé dans les fonctions automatiques peut ainsi être repris sans modification.

La centrale maître enregistre une liste de tous les appareils connus dans le système, ainsi que leur NickName sous forme de fichier de configuration .cs2. Le nom/identificateur peut être prédéfini de manière judicieuse, mais doit pouvoir être modifié par l'utilisateur.

1.3.2.2 - Identifiants de confirmation et identifiants automatiques

peuvent être utilisées pour réaliser de nouvelles fonctions d'automatisation.
Cet identifiant permet d'accéder à la fois à une confirmation et à une fonction d'automatisation. S88 /
Les identifiants de réponse commencent dans l'octet de poids fort de 0 à 63 au maximum, l'octet de poids faible dans la plage respective de 0 à 255. 16 384 contacts de réponse au maximum sont donc possibles par appareil.

La valeur 64 est réservée aux répéteurs SX1.

Les fonctions automatiques commencent dans l'octet de poids fort à 65 de la représentation ASCII de "A". Dans l'octet faible, les valeurs commencent actuellement par convention à ASCII "1", décimal 49 (0x31). (Il s'agit de la représentation ASCII "A1" - donc de la première fonction de mémoire). L'espace entre le répéteur SX1 et les fonctions automatiques est réservé.

Les fonctions d'automatisation déjà existantes sont alors adressées avec la désignation A1 - z8 déjà attribuée. De nouvelles identifications
Tableau de l'adressage qui en résulte.

Identification de l'appareil	Début		Fin		Utilisation
0x0000 -	0x00	0x00	0x3F	0xFF	S88 Répéteur
0xFFFF	0	0	63	255	16384 Répéteur
0x0000 -	0x40	0x00	0x40	0xFF	Réservé SX1 Répéteur 128
0xFFFF	64	0	64	255	Répéteur
0x0000 -	0x41	0x00	0x41		Libre / Réserve
0xFFFF	65	0	65		
0x0000 -	0x41	0x31	0x7F		Mémoire - fonctions automatiques. Commençant par la représentation ascii "A" "1".
0xFFFF	65	49	127		
0x0000 -	0x80	0x00	0xFF	0xFF	Réservé
0xFFFF	128	0	255	255	

1.3.2.3 - Adresses d'automatisation / adresses de confirmation

Ces adresses sont composées de l'identifiant de l'appareil (valeur supérieure) et de l'identifiant de confirmation ou de l'identifiant automatique correspondant. L'identifiant de l'appareil constitue la partie la plus importante.

1.3.3 - Niveaux de conduite dans le système

Les vitesses dans l'ensemble du système sont traitées comme des valeurs de 10 bits. Cette valeur est indépendante de la valeur réelle envoyée à la locomotive (via la voie). La plage de valeurs utilisée doit aller de 0 à 1000, 0 correspondant à une locomotive à l'arrêt, 1000 à la vitesse maximale d'une locomotive.

Des valeurs supérieures à 1000 (jusqu'à 1023) peuvent se produire et ne doivent perturber aucun récepteur. La vitesse de déplacement correspond ici au maximum.

La conversion en pas de vitesses réelles est possible à l'aide des règles de calcul suivantes : pas de vitesse système = 1 + (pas de vitesse voie - 1) * pas de vitesse.

Nombre de vitesses	Incremets
14	77
27	38
28	37
31	33
126	8

Le niveau de vitesse 1 de la voie est donc toujours le niveau de vitesse 1 du système.
Le niveau de vitesse 11 de la voie est pour :

14 niveaux de vitesse : 771
27 niveaux de vitesse : 381
28 niveaux de vitesse : 371
31 niveaux de vitesse : 331
126 niveaux de conduite : 81

Aperçu des valeurs des commandes

Tableau récapitulatif des identificateurs de commande.

Commande	Valeur	Valeur dans CAN-ID	DLC possibles
Commandes système	0x00	0x00	voir ci-dessous
Locomotive Discovery	0x01	0x02	0,1,5,6
MFX Bind	0x02	0x04	6
MFX Verify	0x03	0x06	6,7
Vitesse de la locomotive	0x04	0x08	4,6
Direction de la locomotive	0x05	0x0A	4,5
Fonction de la locomotive	0x06	0x0C	5,6,8
Lire la configuration	0x07	0x0E	6,7
Write Config	0x08	0x10	8
Accessoires de commutation	0x0B	0x16	6,8
Accessoires Config	0x0C	0x18	
S88 Polling (rétroaction)	0x10	0x20	5,6,7,8
Événement S88	0x11	0x22	7,8
Événement SX1	0x12	0x24	5,6
État du logiciel Demande/participant Ping	0x18	0x30	0,5,7,8
Offre de mise à jour	0x19	0x32	8
Lecture des données de configuration	0x1A	0x34	4,8
Bootloader CAN lié, "Service	0x1B	0x36	0,5,6,7,8
Rails de chargement d'amorçage liés, "Service	0x1C	0x38	4,5,6,7,8
Données d'état Configuration	0x1D	0x3A	5,8
Demande de données de configuration,"Data Query".	0x20	0x40	8
Config Data Stream	0x21	0x42	6,7,8
60128 (Connect 6021) Flux de données/ ancienne désignation "6021 adapter	0x22	0x44	5,6

Tableau récapitulatif des commandes système et de la valeur de sous commande

Commande	Sous-CMD Valeur	DLC possibles
Arrêt du système	0x00	5
Système Go	0x01	5
Arrêt du système	0x02	5
Arrêt d'urgence de la locomotive	0x03	5
Arrêt du cycle de la locomotive (terminer)	0x04	5
Locomotive Protocole de données	0x05	6
Temps de commutation du décodeur d'accessoires	0x06	5
Fast Read pour mfx	0x07	6
Activer le protocole de voie	0x08	6
Système MFX Définir le compteur de nouvelles inscriptions	0x09	7
Surcharge du système	0x0A	6
État du système	0x0B	6,8
Identification du système	0x0C	5,7
Mfx Seek	0x30	6,7,8
Réinitialisation du système	0x80	6

2 - Commandes système

Les commandes système concernent directement le processeur de format de voie et déterminent son fonctionnement et ses états. De plus, les états du processeur de format de voie sont signalés. Les commandes incluent, en plus d'un UID, un octet de commande. Cet octet sert à identifier la commande système à exécuter.

2.1 Commande : Arrêt du système

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Arrêt du système (0x00)

Format :

<table>
<tr><th>Prio</th><th>Commande</th><th>Resp.</th><th>Hash</th><th>DLC</th><th>Octet D 0</th><th>Octet D 1</th><th>Octet D 2</th><th>Octet D 3</th><th>Octet D 4</th><th>Octet D 5</th><th>Octet D 6</th><th>Octet D 7</th></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x00</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">MFX-UID</td><td rowspan="2">0x00</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Le processeur de format de voie arrête l'opération sur la voie principale et la voie de programmation. Aucune énergie électrique n'est fournie. Tous les niveaux de vitesse, valeurs de fonction et réglages sont conservés.

Pour un arrêt général affectant tous les processeurs de format de voie, une UID d'appareil cible spéciale est utilisée (0x0000).

Exemples :

- 00004711 5 00 00 00 00 00 Arrêt pour tous
- 00004711 5 43 53 32 08 00 Arrêt pour un participant spécifique (CS2 avec SNr. 08)

Réponse :

- Commande d'origine avec bit de réponse activé

Particularités :

- L'arrêt est toujours déclenché par le processeur de l'interface utilisateur graphique.

- Aucun niveau de vitesse 0 ou arrêt d'urgence n'est envoyé. Après la commande "Go du système", toutes les locomotives reprennent avec leurs anciens réglages ou restent en place. Ce comportement est déterminé par le décodeur.

2.2 Commande : Démarrage du système

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Démarrage du système (0x01)

Format :

<table>
<tr><th>Prio</th><th>Commande</th><th>Resp.</th><th>Hash</th><th>DLC</th><th>Octet D 0</th><th>Octet D 1</th><th>Octet D 2</th><th>Octet D 3</th><th>Octet D 4</th><th>Octet D 5</th><th>Octet D 6</th><th>Octet D 7</th></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x0</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">MFX-UID</td><td rowspan="2">0x01</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Le processeur de format de voie active l'opération et fournit de l'énergie électrique. Tous les niveaux de vitesse et les fonctions éventuellement encore présents ou enregistrés sont renvoyés.

Pour une commande générale de démarrage affectant tous les processeurs de format de voie, une UID d'appareil cible spéciale est utilisée (0x0000).

Exemples :

- 00004711 5 00 00 00 00 01 Démarrage pour tous
- 00004711 5 43 53 32 08 01 Démarrage pour un participant spécifique (CS2 avec SNr. 04)

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- Le démarrage est toujours déclenché par le processeur de l'interface utilisateur graphique.

2.3 Commande : Arrêt du système

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Arrêt du système (0x02)

Format :

<table>
<tr><th>Prio</th><th>Commande</th><th>Resp.</th><th>Hash</th><th>DLC</th><th>Octet D 0</th><th>Octet D 1</th><th>Octet D 2</th><th>Octet D 3</th><th>Octet D 4</th><th>Octet D 5</th><th>Octet D 6</th><th>Octet D 7</th></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x0</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">MFX-UID</td><td rowspan="2">0x02</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Toutes les locomotives reçoivent l'ordre de s'arrêter (vitesse 0), y compris l'inertie de freinage. Le signal numérique reste sur les rails, mais aucune autre commande n'est envoyée sur les rails. L'énergie électrique reste disponible. Utile pour un arrêt contrôlé de l'installation.

Pour une commande générale d'arrêt affectant tous les processeurs de format de voie, une UID d'appareil cible spéciale est utilisée (0x0000).

Exemples :

- 00004711 5 00 00 00 00 02 Arrêt pour tous
- 00004711 5 43 53 32 08 02 Arrêt pour un participant spécifique (CS2 avec SNr. 04)

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- L'arrêt est toujours déclenché par le processeur de l'interface utilisateur graphique.

2.4 Commande : Arrêt d'urgence de la locomotive

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Arrêt d'urgence de la locomotive (0x03)

Format :

<table>
<tr><th>Prio</th><th>Commande</th><th>Resp.</th><th>Hash</th><th>DLC</th><th>Octet D 0</th><th>Octet D 1</th><th>Octet D 2</th><th>Octet D 3</th><th>Octet D 4</th><th>Octet D 5</th><th>Octet D 6</th><th>Octet D 7</th></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x0</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">MFX-UID</td><td rowspan="2">0x03</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Arrêt d'urgence ou arrêt immédiat de la locomotive, selon le protocole de voie. Il faut spécifier une locomotive déjà ciblée par une commande. Si cette locomotive n'est pas déjà dans le cycle, elle ne sera pas prise en compte par cette commande.

Exemples :

- 00004711 5 00 00 00 48 03 Arrêt d'urgence locomotive MM2 72
- 00004711 5 00 00 C0 03 03 Arrêt d'urgence locomotive DCC Adr. 3
- 00004711 5 00 00 40 05 03 Arrêt d'urgence locomotive MFX SID 5

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- L'arrêt d'urgence n'est pas implémenté comme une vitesse, pour éviter toute mauvaise interprétation avec différents protocoles.
- L'arrêt d'urgence est toujours déclenché par le processeur de l'interface utilisateur graphique.
- La première commande intègre la locomotive dans le cycle, et l'arrêt d'urgence est envoyé.

2.5 Commande : Mettre fin au cycle de locomotive

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Arrêt du cycle de locomotive (0x04)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x00</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">MFX-UID</td><td rowspan="2">0x04</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>Haute</td><td></td><td></td><td>Low</td></tr>
</table>

Description :

Supprime la locomotive de la liste de gestion du processeur de format de voie. La locomotive ne reçoit plus de commandes (pas de cycle de rafraîchissement). Elle ne sera à nouveau adressée et recevra des télégrammes de voie qu'après avoir été configurée avec une vitesse ou avoir été adressée par des fonctions.

Exemples :

- 00004711 5 00 00 00 48 04 Mettre fin au cycle MM2 72
- 00004711 5 00 00 C0 03 04 Mettre fin au cycle DCC adr. 3
- 00004711 5 00 00 40 05 04 Mettre fin au cycle MFX SID 5
- 00004711 5 43 53 32 08 04 Mettre fin au cycle de toutes les locomotives dans le processeur de format de voie avec UID=43 53 32 08
- 00004711 5 00 00 00 00 04 Mettre fin au cycle de toutes les locomotives sur tous les processeurs de format de voie (UID 00 00 00 00, diffusion à tous)

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- Toujours déclenché par le processeur d'interface utilisateur graphique (GUI).
- Si l'ID de la locomotive correspond à l'UID du processeur de format de voie, l'ensemble du tableau de gestion des locomotives est effacé.

2.6 Commande : Protocole de données de locomotive

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Protocole de données de locomotive (0x05)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x00</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">6</td><td colspan="4">MFX-UID</td><td rowspan="2">0x05</td><td rowspan="2">Paramètre</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>Haute</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Modifie le sous-protocole de voie spécifique, différent de la valeur par défaut, sans changer le protocole de voie principal, ce qui évite le changement de l'ID de la locomotive. Pour MM2 :

- 0 : MM2 2040, Commande des locomotives à 20 kHz et FDEC à 40 kHz
- 1 : MM2_LOK_20, Commande uniquement des locomotives à 20 kHz
- 2 : MM2_FKT_40, Commande uniquement de FDEC à 40 kHz

Pour DCC :

- 0 : Adresse courte DCC, 28 niveaux de vitesse [=DCC-FS-Default]
- 1 : Adresse courte DCC, 14 niveaux de vitesse
- 2 : Adresse courte DCC, 126 niveaux de vitesse
- 3 : Adresse longue DCC, 28 niveaux de vitesse
- 4 : Adresse longue DCC, 126 niveaux de vitesse

Exemple :

- 00004711 6 00 00 C0 03 05 02 Commande DCC adr. 03 avec 126 niveaux de vitesse
- (Note : Correction apportée pour "DCC Adr 03 avec 126 Fahrstufen ansteuern" en "Commande DCC adr. 03 avec 126 niveaux de vitesse")

Réponse :

- Commande d'origine avec bit de réponse activé.
- Aucune réponse envoyée en cas d'ID de locomotive non valide (ni DCC, ni MM2).

Particularités :

- Toujours déclenché par le processeur d'interface utilisateur graphique (GUI).
- Le premier ordre prend le décodeur de locomotive/fonction dans le cycle.

2.7 Commande : Définir le temps de commutation du décodeur d'accessoire

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Temps de commutation (0x06)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x00</td><td rowspan="2">0x00</td><td rowspan="2">1</td><td rowspan="2"></td><td rowspan="2">7</td><td colspan="4">ID de périphérique cible (Ziel Geräte-UID)</td><td rowspan="2">0x06</td><td colspan="2">Temps</td><td rowspan="2"></td></tr>
<tr><td>Haute</td><td></td><td></td><td>Low</td><td>High</td><td>Low</td></tr>
</table>

Description : Définit le temps par défaut pour la commutation des décodeurs d'accessoire. Le temps peut également être spécifié lors de la commande de commutation. Le temps est en incrément de 10 ms, avec un maximum de 163 secondes (environ 2 minutes 45 secondes). La valeur 0 rétablit le temps de commutation par défaut.

Exemple :

- 00004711 7 00 00 00 00 06 0A Définir le temps de commutation à 100 ms
- (Note : Correction apportée pour "Schaltzeit 100 ms festlegen" en "Définir le temps de commutation à 100 ms")

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- Toujours déclenché par le processeur d'interface utilisateur graphique (GUI).

2.8 Commande : Lecture rapide pour adresse SID MFX

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Lecture rapide (0x07)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x00</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">7</td><td colspan="4">UID de l'appareil expéditeur</td><td rowspan="2">0x07</td><td colspan="2">MFX-SID</td><td rowspan="2"></td></tr>
<tr><td>Haute</td><td></td><td></td><td>Low</td><td>High</td><td>Low</td></tr>
</table>

Description :

Permet de supprimer le délai entre les commandes de lecture pour cette adresse SID MFX à partir de la version 2 de GFP. La fonction de Lecture rapide est seulement compatible avec les décodeurs de génération plus récente. Les décodeurs MFX de première génération ne sont pas conçus pour cela et pourraient être endommagés. La Lecture rapide ne peut être activée qu'après une vérification de plausibilité par le GFP. L'UID pour cette SID doit être dans la plage des décodeurs Märklin MFX (commençant par 0x7F). Cela est réalisé par une vérification MFX. Cette vérification doit être effectuée avant cette commande et répondre positivement.

Exemple :

- 00004711 7 00 00 00 00 07 7F 02 Activer la Lecture rapide pour l'adresse SID mfx 7F 02

Réponse :

- Toujours : Commande d'origine avec bit de réponse activé.

Particularités :

- Dans la version 1.0, la commande est rejetée.
- Toujours déclenché par le processeur d'interface utilisateur graphique (GUI).

2.9 Commande : Activer ou désactiver le protocole de voie

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Protocole de voie (0x08)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">0x0</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">6</td><td colspan="4">MFX-UID</td><td rowspan="2">0x08</td><td rowspan="2">Paramètre</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>Haute</td><td></td><td></td><td>Low</td></tr>
</table>

Description : Active ou désactive les protocoles correspondants sur la voie. Le paramètre est bit codé pour activer les protocoles correspondants sur la voie. Un bit activé libère le protocole correspondant, un bit désactivé le supprime. Après la réinitialisation, tous les protocoles sont activés.

Bit# Protocole :

Bit#	Protocole
0	MM2
1	MFX
2	DCC
3	Res
4	Res
5	Res
6	Res
7	Res

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- GFP prend en charge uniquement MM2, MFX et DCC.
- Lorsque MFX est désactivé, aucune locomotive MFX n'est recherchée.
- Les paramètres ne sont pas sauvegardés dans le processeur de format de voie. Ils doivent être configurés à chaque démarrage.

2.10 Commande : Réinitialiser le compteur de réenregistrement MFX

Identification :

- Commande système (0x00, dans CAN-ID : 0x00)

Sous-commande :

- Compteur de réenregistrement (0x09)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">Message Prio</td><td rowspan="2">0x00</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">7</td><td colspan="4">MFX-UID</td><td rowspan="2">0x09</td><td colspan="2">Compteur de réenregistrement</td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td><td>High</td><td>Low</td></tr>
</table>

Description : Modifier le compteur de réenregistrement du processeur de format de voie pour le sous-système MFX.

Exemple :

- 00004711 7 43 53 32 08 09 00 02 Réinitialiser le compteur de réenregistrement à 2.

Réponse :

- Commande d'origine avec bit de réponse activé.

Particularités :

- Toujours déclenché par le processeur de l'interface utilisateur graphique.

3 - Administration

Les instructions de gestion servent à commander les récepteurs auxquels le processeur de format de voie s'adresse. Il ne devrait pas y avoir ici d'instructions qui influencent le comportement du processeur de format de voie.

3.1 - Ordre de mission : Locomotive Discovery

Code d'identification :

Discovery (0x01, dans CAN-ID: 0x02)

Format :

Identifiant du message 29 bits				Long Data	Data							
Priorité	Commande	Reponse	Hash	Length	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Priorité du message	Commande	Commande (0) ou réponse (1)	Résolution des collisions	Nombre de données en bytes 0 à 8	Données	...	...	...				
0x00	0x01	0	0xXXXX	1	Identification du protocole - 0x20 pour MFX							
0x00	0x01	0 ou 1	0xXXXX	5	MFX - UID				Range identification du protocole			
0x00	0x01	0	0xXXXX	6	MFX - UID				Range identification du protocole	ASK		

Description :

Recherche de locomotives sur la voie. La voie est déterminée par l'identifiant du protocole.

Demandes :

Forme 1 : DLC = 0 : Démarrer "Détecter tous les protocoles". (plus tard >= V2.0)

Forme 2 : DLC = 1 : Reconnaissance par protocole de données, protocole - l'identifiant détermine le protocole de données.

Le processeur de format de voie fait la découverte MFX / la reconnaissance de locomotive de manière autonome. Les étapes intermédiaires de la découverte sont communiquées sous la forme 3.

Un seul décodeur est reconnu par processus de reconnaissance.

Réponse :

Positif avec DLC=5 et adresse trouvée (jusqu'à présent)
Négatif : DLC=0

Forme 3 : DLC = 5 :

1. Single - MFX - Discovery : Envoi de la requête avec Range (=Info longueur des bits) (Réalisé dans V1.0)

2. Autre reconnaissance : L'adresse et le protocole sont extraits du Loc-ID. Avec ces données, on essaie de reconnaître cette adresse sur la voie.

Réponses :

Pour indiquer la reconnaissance MFX, les étapes intermédiaires sont également communiquées lors d'un cycle commandé par le processeur de format de voie. Seule une plage = 32 détermine l'UID complet du décodeur.

Pour le debug : DLC = 6 : Single - MFX - Discovery : réponse avec Range et rapport ASK.

Demande de voie de programmation :		
Protocole	Gamme/ Protocole-Identification	Remarque
MFX :	0-32	=Range. Demande de reconnaissance de la locomotive sur la voie deprogrammation. Si la locomotive répond, le bit de réponse est mis à 1. Cas particulier : Range=0 : une réinscription forcée d'un décodeur sur le PGL.
MM2 :	33	Reconnaissance de la locomotive MM2 sur la voie de programmation.Dans la réponse, l'adresse MM2 est signalée. (20 kHz)
MM2 :	34	Reconnaissance de la locomotive MM2 sur la voie de programmation.L'adresse MM2 est signalée dans la réponse. (40 kHz) (>V3.0)
DCC :	35	Lecture de l'adresse DDC sur la voie de programmation. Provoque le même processus que 36, réponse soit 35 soit 36. Dans la réponse, l'adresse DCC courte est signalée.
DCC	36	Lecture de l'adresse DDC sur la voie de programmation. Provoque le même processus que 35, réponse soit 35 soit 36.Dans la réponse, l'adresse DCC longue est signalée.
DCC :	37	Détection DCC (algorithme d'entrelacement d'intervalles avec DCC-K=127Adr). L'adresse DCC est signalée dans la réponse. (>V3.0)

SX1 :	38	Lecture de l'adresse SX1 sur la voie de programmation.L'adresse SX1 est signalée dans la réponse. (>V3.0)
SX1 :	39	Reconnaissance (algorithme d'emboîtement d'intervalles avec Adr 0-99)L'adresse SX1 est signalée dans la réponse. (>V3.0)

Demande de voie principale :		Range = valeur mod 64.
MFX :	64-96	Demande de reconnaissance de la locomotive sur la voie principale. Si la locomotive répond, le bit de réponse est mis à 1
MM2 :	98	Non valide, car impossible d'un point de vue matériel
DCC :	99	Non valide, car impossible d'un point de vue matériel
DCC :	100	Non valide, car impossible d'un point de vue matériel
SX1 :	101	Non valide, car impossible d'un point de vue matériel
SX1 :	102	Non valide, car impossible d'un point de vue matériel

Exemple :

00024711 1 20	Cycle complet de découverte mfx
00024711 5 FF FF FF 00	Range 0 MFX Discovery
00024711 1 21	MM2 Découverte

Particularités :

Est toujours déclenché par le processeur de l'interface utilisateur graphique. Le déclenchement d'un Discovery ne devrait être effectué que par le pupitre de commande maître. Sinon, il faut prendre des dispositions pour le déclenchement simultané de dicoverys.

Les réponses à une découverte doivent et peuvent être reçues par tous les participants et également évaluées en conséquence.

Mfx Discovery ne fait pas de distinction entre la voie principale et la voie de programmation.

Si une locomotive MFX est recherchée selon la forme 1, le cycle complet de 32 étapes est effectué en cas de réponse.

Le rapport ASK est une valeur caractéristique de la qualité des signaux de retour MFX.

Lors d'un cycle complet, le dernier message est envoyé avec et sans rapport ASK. Il s'agit d'une commande séquentielle en cours d'exécution.

Instruction de programmation. Celles-ci ne sont pas stockées temporairement dans une file d'attente d'instructions. Ce n'est qu'après une réponse du processeur de format de voie, l'instruction de programmation suivante peut être demandée. Une seule instruction de programmation est exécutée simultanément par le processeur de format de voie.

Commentaires et interprétations :

Tout ce qui précède nous semble bien incompréhensible pour une fonctionnalité qui est pourtant l'une des plus intéressantes en MFX, à savoir la reconnaissance automatique des locomotives fonctionnant sous ce protocole.

Locomotive Discovery est la fonction à appeler pour activer le processus de découverte du MFX.

Comme indiqué au tableau format, la fonction en C++ s'écrit comme suit :

```
out.command = 0x01;
out.length = 1;
out.data[0] = 0x20; // Pour MFX
```

La commande est 0x01, DLC = 1 et le seul byte de données dans data[0] est 0x20 pour un process de découverte ne concernant que ce protocole.
Cette méthode n'est appelée qu'une seule fois pour l'ensemble d'une session. Elle va tourner en boucle sur la Gleisbox. Elle appelle tous les UID possibles en MFX et lorsqu'un identifiant est trouvé, adresse une réponse sous ce format :

```
CAN_Command     : 0x01
Frame length : 6
Response            : 1
Data[0] 0x73 Data[1] 0xF6 Data[2] 0x88 Data[3] 0x34 Data[4] 0x00 Data[5] 0x05
```

L'UID est contenu dans les quatre premiers bytes soit dans cet exemple 0x73F68834 et les deux bytes suivants nous fournissent le Loc-ID 0x0005.

Dans le cas où plusieurs locomotives sont sur les rails, il ne sera renvoyé de réponse que pour une seule locomotive.

Pour enregistrer plusieurs locomotives, il faut les poser les unes après les autres sur les rails. Une locomotive déjà identifiée peut rester sur les rails.

3.2 - Commande : MFX Bind

Code d'identification : Bind (0x02, dans CAN-ID : 0x04)

Format :

Prio	Command e	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Message Prio	0x02	0|1		6	MFX-UID				MFX-SID			
					High			Low	Haute	Low		

Description :

Attribuer le SID MFX à un décodeur MFX avec l'UID MFX au moyen de mfx-BIND, enregistrer une locomotive mfx trouvée par Discovery. Le décodeur peut ensuite être adressé avec un Loc-ID.

Pour les décodeurs mfx, une attribution d'adresse automatisée est réalisée. Dans l'interface graphique, ils sont sélectionnés par le nom de la locomotive et sont également conduits par cette identification. Une adresse de rail n'est pas visible ici. L'appareil de commande utilise cependant une adresse pour la commande des décodeurs.

En outre, chaque décodeur a un UID unique avec lequel il se fait connaître dans le système. Cet identifiant est utilisé pour la procédure d'inscription (voir Discovery). La commande Bind permet d'attribuer au décodeur une adresse de rail plus courte. Grâce à cette adresse de rail, le décodeur peut être commandé et il ne participe plus à la procédure d'inscription.

Exemple :
00044711 6 FF FA 8C 43 00 05 Bind UID : FF FA 8C 43 sur SID : 05
00054711 6 FF FA 8C 43 00 05 Réponse

Réponse :

Instruction initiale avec bit de réponse activé. La réponse indique la fin de l'exécution.
Particularités :

Valable uniquement pour MFX.

Est toujours déclenché par le processeur de l'interface utilisateur graphique. Le déclenchement de Bind ne devrait être effectué que par le pupitre de commande maître. Sinon, il faut prendre des dispositions pour le déclenchement simultané de Binds.

Les réponses à un bind doivent et peuvent être reçues par tous les participants et également évaluées en conséquence.

Il s'agit d'une commande de programmation séquentielle en cours de traitement. Celles-ci ne sont pas stockées temporairement dans une file d'attente d'instructions. L'instruction de programmation suivante ne peut être demandée qu'après une réponse du processeur de format de voie. Une seule instruction de programmation est traitée simultanément par le processeur de format de voie.

3.3 - Commande : MFX Verify

Code d'identification : Verify (0x03, dans CAN-ID : 0x06)

Format :

Prio	Commande	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Message Prio	0x03	0|1		6	MFX-UID				MFX-SID			
					High			Low	High	Low		
Message Prio	0x03	1		7	MFX-UID				MFX-SID		ASK-Rapport	
					High			Low	High	Low		

Description :

DLC = 6 : Demande de vérification de la présence d'une locomotive sous la combinaison MFX-UID / MFX-SID.

DLC = 7 : Si cette combinaison est présente, une confirmation est envoyée avec MFX-UID et MFX-SID ainsi que le rapport ASK (qualité de la réponse). Si la réponse est négative, le SID est fixé à 0x0000.

Exemple :
00064711 6 FF FA 8C 43 00 05 Verify FF FA 8C 43 sur SID 05
00074711 6 FF FA 8C 43 00 00 Réponse négative
00074711 7 FF FA 8C 43 00 05 D1 Réponse positive avec rapport ASK 0xD1 Réponse :

DLC = 6 : Si la combinaison n'existe pas, le SID est fixé à 0x0000 dans la réponse avec le bit de réponse activé.

DLC = 7 : Si la combinaison est présente, la réponse est l'instruction d'origine avec le bit de réponse activé et le rapport ASK dans l'octet D 6.

Particularités : Valable uniquement pour MFX.

Si un décodeur MFX lié reçoit une combinaison erronée d'UID MFX et de SID MFX, un UnBIND est déclenché dans le décodeur de locomotive avec le SID MFX correspondant.

Est toujours déclenché par un appareil de commande. Le déclenchement de Verify ne devrait être effectué que par la console de paramétrage maître. Sinon, il faut prendre des dispositions pour le déclenchement simultané de Verifys.

Les réponses à une vérification doivent et peuvent être reçues par tous les participants et également évaluées en conséquence.

Il s'agit d'une commande de programmation séquentielle en cours de traitement. Celles-ci ne sont pas stockées temporairement dans une file d'attente d'instructions. L'instruction de programmation suivante ne peut être demandée qu'après une réponse du processeur de format de voie. Une seule instruction de programmation est traitée simultanément par le processeur de format de voie.

A noter que la vérification peut se faire avec l'UID à 0, par exemple avec data[5] = 0x05 pour MFX-SID :

CAN_Command : 0x03
Frame len 6
Response : 0
Data : [0]0x00 [1]0x00 [2]0x00 [3]0x00 [4]0x00 [5]0x05

3.4 - Commande : LocVitesse

Code d'identification : Vitesse locomotive (0x04, dans CAN-ID : 0x08)

Format :

Prio	Commande	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Message Prio	0x04	0|1		4	Loc-ID							
					High			Low				
Message Prio	0x04	0|1		6	Loc-ID				Vitesse			
					High			Low	High	Low		

Description :

Il s'agit de l'ordre de marche pour les locomotives. Si le Loc-ID est connu, une locomotive peut être conduite immédiatement. En analysant les ordres de marche (réponses), un affichage peut être actualisé en permanence.

Les vitesses dans l'ensemble du système sont traitées comme des valeurs de 10 bits. Cette valeur est indépendante de la valeur réelle envoyée à la locomotive (par la voie). La plage de valeurs utilisée doit aller de 0 à 1000, 0 correspondant à une locomotive à l'arrêt, 1000 à la vitesse maximale d'une locomotive.

Des valeurs supérieures à 1000 (jusqu'à 1023) peuvent être utilisées et ne doivent pas perturber un récepteur. La vitesse de marche correspond ici au maximum.

DLC = 6, définition de la vitesse :
La locomotive avec LOC-ID est pilotée à la vitesse. Vitesse dans la plage de 0 à 1024 (10 bits). Pour tous les protocoles, la vitesse est convertie en vitesse réelle possible. Vitesse 0 est locomotive - ordre d'arrêt avec temporisation de démarrage et de freinage réglée (pas arrêt d'urgence).

DLC = 4, demande du pas de vitesse :
Demande de la vitesse actuelle en l'absence de valeur de vitesse (DLC = 4) Exemple :
00084711 6 00 00 08 03 03 20 00084

Exemple :

00084711 6 00 00 08 03 03 20 Locomotive vitesse SX1 Adr 3, V=0x0320=800 de 1024
00084711 6 00 00 08 03 00 A0 Locomotive vitesse SX1 Adr 3, V=0x00A0=10 de 1024
00084711 6 00 00 40 01 03 20 Locomotive vitesse mfx Adr 1, V=0x0320=800 de 1024
00084711 6 00 00 C0 03 01 20 Locomotive vitesse DCC Adr 3 V=0x0120=288 de 1024
00084711 6 00 00 C0 03 00 A0 Locomotive vitesse DCC Adr 3 V=0x00A0=10 de 1024

00084711 6 00 00 D6 C6 00 A0 Locomotive vitesse DCC Adr 5830 V=0x00A0=10 de 1024

Adresse DCC = 5830 = 0xD6C6 – 0xC000 (54982 - 49152)

Réponse :

Définition d'une vitesse :
Commande initiale avec bit de réponse activé.
Lecture de la vitesse :
Réponse sous forme de définition de la vitesse si la locomotive est connue.
Sinon, information de vitesse manquante (demande initiale).
Particularités de la procédure :
La première commande intègre la locomotive/le décodeur de fonctions dans le cycle.

3.5 - Commande : SensLoc

Code d'identification Direction de la locomotive (0x05, dans CAN-ID : 0x0A)

Format :

Prio	Commande	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Message Prio	0x05	0		4	Loc-ID							
					High			Low				
Message Prio	0x05	0|1		5	Loc-ID				Sens			
					High			Low				

Description de la fonction:

Interroger ou définir le sens de marche d'une locomotive.

DLC = 4, demande de la direction :
Demander la direction actuelle. Réponse avec 5 octets sous forme de " Définition de la direction ".

DLC = 5, Définition de la direction :
Modifier le sens de marche selon le paramètre "Direction". En cas de modification du sens, le processeur de format de voie assure la vitesse 0, la locomotive est freinée avec le décodeur "ABV". Si le sens n'est pas modifié, le pas de vitesse n'est pas modifié.

Signification du paramètre Direction :
0 = sens de marche reste
1 = sens de marche avant
2 = sens de marche arrière
3 = inverser le sens de marche Reste : le sens reste inchangé

Réponse :
Définition de la direction :
Commande initiale avec bit de réponse activé.
Demande de la direction :
Réponse sous la forme "définir la direction". Une réponse est donnée dans tous les cas, même si la locomotive n'est pas connue.

Particularités de la commande :
La première commande intègre la locomotive/le décodeur de fonctions dans le cycle.
Selon le décodeur de locomotive, il se peut que l'inversion du sens ne se fasse qu'à l'arrêt. La nouvelle direction est en tout cas envoyée par le processeur de format de voie.
Est toujours déclenché par le processeur de l'interface utilisateur graphique.

3.6 - Commande : FonctionLoc

Code d'identification : Vitesse locomotive (0x06, dans CAN-ID : 0x0C)

Format :

<table>
<tr><td>Prio</td><td>Commande</td><td>Resp.</td><td>Hash</td><td>DLC</td><td>Octet D 0</td><td>Octet D 1</td><td>Octet D 2</td><td>Octet D 3</td><td>Octet D 4</td><td>Octet D 5</td><td>Octet D 6</td><td>Octet D 7</td></tr>
<tr><td>2+2 bits</td><td>8 bits</td><td>1 bit</td><td>16 bits</td><td>4 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td><td>8 bits</td></tr>
<tr><td rowspan="2">Message Prio</td><td rowspan="2">0x06</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">5</td><td colspan="4">Loc-ID</td><td rowspan="2">Fonction</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
<tr><td rowspan="2">Message Prio</td><td rowspan="2">0x06</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">6</td><td colspan="4">Loc-ID</td><td rowspan="2">Fonction</td><td rowspan="2">Valeur</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
<tr><td rowspan="2">Message Prio</td><td rowspan="2">0x06</td><td rowspan="2">0|1</td><td rowspan="2"></td><td rowspan="2">8</td><td colspan="4">Loc-ID</td><td rowspan="2">Fonction</td><td rowspan="2">Valeur</td><td colspan="2" rowspan="2">Valeur de la fonction</td></tr>
<tr><td>High</td><td></td><td></td><td>Low</td></tr>
</table>

Description :

Il s'agit de l'ordre de marche pour les locomotives. Si le Loc-ID est connu, une locomotive peut être conduite immédiatement. En analysant les ordres de marche (réponses), un affichage peut être actualisé en permanence.

Les vitesses dans l'ensemble du système sont traitées comme des valeurs de 10 bits. Cette valeur est indépendante de la valeur réelle envoyée à la locomotive (par la voie). La plage de valeurs utilisée doit aller de 0 à 1000, 0 correspondant à une locomotive à l'arrêt, 1000 à la vitesse maximale d'une locomotive.

Des valeurs s

Déclencher / désactiver la fonction de la locomotive ou l'interroger.

La fonction se trouve dans la plage 0-31. 0 correspond à F0, 31 correspond à F31. L'état d'autres fonctions n'est pas enregistré dans le processeur de format de voie.

Valeur dans la plage de 0 à 31, 0 = désactivé, 1 - 31 activé. Pour les protocoles qui supportent une valeur de fonction, celle-ci est envoyée au décodeur. Les valeurs de fonction ne sont pas enregistrées dans le processeur de format de voie, seul l'état est enregistré.

DLC = 5 : Demande d'état de la fonction.
Demande de l'état d'une fonction. Seul l'état actif ou inactif est fourni, pas l'état de variation éventuel du processeur de format de voie, pas celui de la demande du décodeur.

DLC = 6 : Activation d'une fonction.
Activation d'une fonction. Selon le protocole de voie, la fonction correspondante est activée ou commandée avec la valeur de variation correspondante. La plage valable d'une valeur de fonction dépend des décodeurs de locomotive et du protocole de voie.

Spécificités :

La première commande intègre la locomotive/le décodeur de fonctions dans le cycle.

Les fonctions de variation et la valeur de variation ne sont pas conservées dans le processeur de format de voie (mémoire). Tant que la valeur de variation doit être active, elle est signalée comme active (1).

Est toujours déclenché par le processeur de l'interface utilisateur graphique.

3.7 - Commande : Read Config

Code d'identification : Vitesse locomotive (0x07, dans CAN-ID : 0x0E)

Format :

Prio	Commande	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
0x00	0x07	0		7	Loc-ID				Index CV (6 bits) Numéro CV (10 bits)		Nombre	
					High			Low				
0x00	0x07	1		7	Loc-ID				Index CV (6 bits) Numéro CV (10 bits)		Valeur	
					High			Low				
0x00	0x07	1		6	Loc-ID				Index CV (6 bits) Numéro CV (10 bits)			
					High			Low				

Description :

Lecture de valeurs à partir de décodeurs pouvant être confirmés.

Lors d'une demande, le numéro de CV et l'index de départ sont indiqués. Le nombre d'octets à lire détermine combien doivent être lus. Une demande peut déclencher plusieurs réponses en fonction du nombre d'octets à lire.

Loc-ID détermine le protocole et l'adresse du décodeur.

Le numéro CV détermine quelle variable de configuration doit être lue. Un total de 1024 adresses est possible

Le numéro CV se trouve dans l'octet D 5 et les 2 bits de poids faible de l'octet D 4.

L'index CV détermine l'index du numéro CV à lire. L'index CV n'est autorisé que pour Mfx. L'index CV se trouve dans les 6 bits de poids fort de l'octet D 4.

Lors de la réponse, une valeur est envoyée octet par octet. Le numéro CV et l'index CV déterminent de quel octet il s'agit. Le caractère de remplacement Nombre contient maintenant la valeur lue.

Si aucune valeur n'a pu être lue, cela est signalé par un accusé de réception négatif avec DLC = 6, c'est-à-dire une valeur manquante.

Plage de valeurs et comportement typiques du protocole :

DCC :

L'index de la CV n'est pas traité. Le numéro CV se situe dans la plage 1 - 1024.

Le paramètre "Nombre" détermine le nombre d'octets à lire à partir du numéro de CV de départ indiqué. La valeur "00" pour "Nombre" lit 256 octets dans le décodeur.

L'octet lu est ajouté à la réponse. Le numéro CV est actualisé dans la réponse. Si aucune valeur n'a pu être lue, une trame est émise avec DLC=6 et l'octet D 6 manquant.

Le décodeur ne peut être lu que sur la voie de programmation.

MFX :

Le numéro CV se situe dans la plage entre 1 et 1024.

L'index CV est pris en compte et se situe dans la plage entre 0 et 63.

Le paramètre "Nombre" détermine le nombre d'octets à lire à partir du début indiqué - indice CV.

Le paramètre Numéro CV reste constant. Un maximum de 63 octets peut être lu par instruction. Le processeur de format de voie décompose la demande de lecture en instructions de lecture multi-octets et lit au maximum 4 octets simultanément dans le décodeur.

L'octet lu est ajouté à la réponse. L'index CV est actualisé dans la réponse. Si aucune valeur n'a pu être lue, une trame avec DLC=6 et l'octet D 6 manquant est émise.

Le décodeur peut être lu sur la voie principale et sur la voie de programmation. Il n'y a pas de distinction.

SX1 :

L'index CV n'est pas traité. Le numéro CV se situe dans la plage 1 - 5.

Le paramètre "Nombre" détermine le nombre d'octets à lire à partir du numéro de CV de départ indiqué "Numéro de CV". Il est possible de lire au maximum 5 octets.

L'octet lu est ajouté à la réponse. Le numéro de CV est actualisé dans la réponse. Si aucune valeur n'a pu être lue, une trame avec DLC=6 et l'octet D 6 manquant est émise.

Comme SX1 ne connaît pas la numérotation des valeurs de configuration, la mise en œuvre suivante est effectuée :

Numéro CV	Signification
1	Adresse
2	Vitesse maximale
3	Accélération
4	Largeur de l'impulsion du moteur
5	1 / 2 - Sections d'arrêt

Un décodeur ne peut être lu que sur la voie de programmation. MM2 :
Les locomotives MM2 ne peuvent pas être lues.

Caractéristiques particulières :

Est toujours déclenché par le processeur Graphical User Interface. L'ordre n'inclut pas la locomotive/le décodeur de fonction dans le cycle.

Selon le protocole, uniquement possible sur la voie de programmation.

Cette commande est une commande de programmation séquentielle en cours de traitement. Elle n'est pas stockée temporairement dans une file d'attente d'instructions. Ce n'est qu'après une réponse du processeur de format de voie que l'instruction de programmation suivante peut être demandée. Une seule instruction de programmation est traitée simultanément par le processeur de format de voie.

3.8 Commande : Écrire Config

Identification : Écrire Config (0x08, dans CAN-ID : 0x10)

Format :

Prio	Commande	Resp.	Hash	DLC	Octet D 0	Octet D 1	Octet D 2	Octet D 3	Octet D 4	Octet D 5	Octet D 6	Octet D 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
0x01	0x08	0		8	Loc-ID				CV-Index (6 Bit) CV-Numéro (10 Bit)		Valeur	Ctrl(2Bit)
					Haute			Low				
0x01	0x08	1		8	Loc-ID				CV-Index (6 Bit) CV-Numéro (10 Bit)		Valeur	Rslt(2Bit)
					Haute			Low				

Description :

Écriture de valeurs de CV dans un décodeur programmable.

En fonction du protocole du décodeur à programmer, différents paramètres sont possibles. L'écriture des valeurs de CV est, si possible, vérifiée par une lecture ultérieure. Dans le cadre de confirmation, le résultat de la vérification est communiqué. (> V3.0)

Loc-ID contient le protocole et l'adresse du décodeur à programmer.

CV-Numéro détermine quelle variable de configuration doit être modifiée. Il est possible d'avoir un total de 1024 adresses. CV-Numéro est situé dans D-Byte 5 et les 2 bits de poids faible de D-Byte 4.

CV-Index détermine un index possible du CV-Numéro à modifier. CV-Index est seulement valable pour Mfx. CV-Index est situé dans les 6 bits de poids fort de D-Byte 4.

Le paramètre Valeur contient l'octet à écrire. Pour le type de programmation "DCC-Bitprogrammierung", l'octet de données contient l'information dans le format DCC correspondant : 1111DBBB où D représente la valeur du bit et BBB représente la position du bit.

La signification du dernier octet dans le télégramme est différente pour la demande et la confirmation :

Demande : Ctrl (Bit 8 & Bit 7) contient des instructions pour la commande.

- Bit 8 : Distinction entre voie principale (valeur = 1) ou voie de programmation (valeur = 0)
- Bit 7 : Écriture multioctets. D'autres commandes d'écriture suivront.
- Bit 6 : Type de programmation DCC 1 (DCC registre ou direct/bit programmation)
- Bit 5 : Type de programmation DCC 2
- Anz (Bits 4 à 0) est réservé.

Sélection du type de programmation DCC :

DCC1	DCC2	Type
0	0	Programmation directe
0	1	Programmation par registre
1	0	Programmation par bit
1	1	Réservé

Confirmation : Dans Result (Bit 8 & Bit 7) figurent les résultats de l'écriture et de la vérification :

- Bit 8 : Écriture réussie confirmée par le contrôleur.
- Bit 7 : Vérification réussie.

Description complémentaire :

Selon le type de décodeur, des résultats négatifs peuvent être considérés comme réussis. Par exemple, si le décodeur ne peut ni lire ni confirmer les valeurs.

Comportement et plages de valeurs spécifiques aux protocoles :
DCC :

- CV-Index : Non traité.
- CV-Numéro : Entre 1 et 1024.
- Écriture : Sur la voie de programmation et la voie principale.
- Le décodeur peut être entièrement programmé sur la voie de programmation.
- Les paramètres manipulables par "POM" peuvent également être modifiés sur la voie principale.

MFX :

- CV-Numéro : Entre 1 et 1024.
- CV-Index : Traité et entre 1 et 63.
- L'écriture se fait principalement sur la voie principale et la voie de programmation.

SX1 :

- CV-Index : Non traité.
- CV-Numéro : Entre 1 et 5.
- Un décodeur ne peut être programmé que sur la voie de programmation.
- Les paramètres n'ont pas de numéros dans SX1. La correspondance est la suivante :

CV-Numéro	Signification
1	Adresse
2	Vitesse maximale
3	Accélération
4	Largeur d'impulsion du moteur
5	Sections d'arrêt 1/2

MM2 :

- CV-Index : Non traité.
- CV-Numéro : Entre 1 et 256.
- La programmation ne peut se faire que sur la voie de programmation.
- L'adresse MM2 spécifiée dans Loc-ID définit l'adresse de programmation "MM2".
- Un décodeur MM2 programmable peut être programmé soit sous son propre numéro d'adresse soit sous l'adresse 80. Ceci doit être pris en compte sur la voie principale.

Particularités :

- Cette commande est toujours déclenchée par le processeur de l'interface graphique.
- La commande ne prend pas en compte les décodeurs de locomotives/fonctions dans le cycle.
- Pour MM2, cette adresse détermine l'adresse de programmation sous laquelle la programmation se déroule, y compris les adresses différentes de 80.
- Cette commande est un ordre de programmation séquentiel en cours d'exécution. Ils ne sont pas mis en file d'attente de commandes. Une nouvelle commande de programmation ne peut être demandée qu'après une réponse du processeur de format de voie.
- Un seul ordre de programmation est traité simultanément par le processeur de format de voie.

6 - Autres commandes

6.1 – Commande à effectuer : État du logiciel Demande / Participant Ping

Code d'identification :

Etat du logiciel / participant Ping (0x18, dans CAN-ID : 0x30)

Format :

Le format des messages CAN pour l'appareil qui envoie le message est :

Identifiant du message 29 bits				DLC	Data							
Priorité	Commande	Réponse	Hash	Length	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Priorité du message	Commande	Commande (0) ou réponse (1)	Résolution des collisions	Nombre de données en bytes 0 à 8	Données	…						
0x00	0x18	0	(hash)	0								

Le champ réponse est bien sûr à 0 (false), aucune data n'est envoyée donc DLC = 0.

Voici la structure type d'un message de demande :

```
17:24:33.285 -> Frame ID : 0x308B23
17:24:33.285 -> Priorité = 0x00
17:24:33.285 -> CAN_Command = 0x18
17:24:33.285 -> Hash = 0x8B23
17:24:33.285 -> Frame len 0
17:24:33.285 -> Response 0
17:24:33.285 -> Data :
```

Description :

Ce message est envoyé sur le bus CAN. Chaque appareil répond avec les données correspondantes. On obtient ainsi la configuration de tous les participants accessibles sur le bus CAN.

Lors de la réponse, l'UID est remplacé par celui de l'appareil qui répond. Ainsi, le processeur de l'interface utilisateur graphique peut déterminer quels appareils sont connectés.

Voici un exemple de message « retour » :

Identifiant du message 29 bits				Long Data	Data							
Priorité	Commande	Réponse	Hash	Length	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
2+2 bits	8 bits	1 bit	16 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Priorité du message	Commande	Commande (0) ou réponse (1)	Résolution des collisions	Nombre de données en bytes 0 à 8	UID de l'appareil expéditeur				Numéro de version du logiciel		Identification de l'appareil	
0x313363				0x08	0x4746A475				0x012F		0x0011	
0x00	0x18	1	0x3363	8	0x47	0x46	0xA4	0x75	0x01	0x2F	0x00	0x11

```
17:24:33.285 -> Frame ID : 0x313363
17:24:33.318 -> Priorité = 0x00
17:24:33.318 -> CAN_Command = 0x18
17:24:33.318 -> Hash = 0x3363
17:24:33.318 -> Frame len 8
17:24:33.318 -> Response 1
17:24:33.318 -> Data : 0x47 0x46 0xA4 0x75 0x01 0x2F 0x00 0x11
```

Lors de la réponse, l'UID est remplacé par celui de l'appareil qui répond. Ainsi, le processeur de l'interface utilisateur graphique peut déterminer quels appareils sont connectés.

Le numéro de version est un identifiant de la version du logiciel.

Dans l'octet 6 et l'octet 7 (DB), l'information sur le type de l'appareil est codée en big-endian. Actuellement, les identifiants d'appareils suivants sont définis :

Identification de l'appareil	Appareil
0x00 0x00	Format de la voie Processeur 60213,60214 / Booster 60173, 60174
0x00 0x10	Boîte de voies 60112 et 60113
0x00 0x20	Connect 6021 Réf. 60128
0x00 0x30	MS 2 60653, Txxxxx
0xFF 0xE0	Périphériques sans fil
0xFF 0xFF	CS2-GUI (maître)

Dans l'exemple ci-dessus, l'identification de l'appareil est 0x11 qui ne figure pas dans ce tableau. Il s'agit d'une boite de voie (ref 60116) qui n'existait pas au moment de la rédaction du document initial.

Particularités :

Seules les unités de commande répondent. Pas les décodeurs de locomotive.

Précision importante car il n'est pas possible par cette méthode de connaître les locomotives présentes sur les voies.

Avec une MS2, cette commande de scan des appareils présents est répétée toutes les 12 secondes environ.