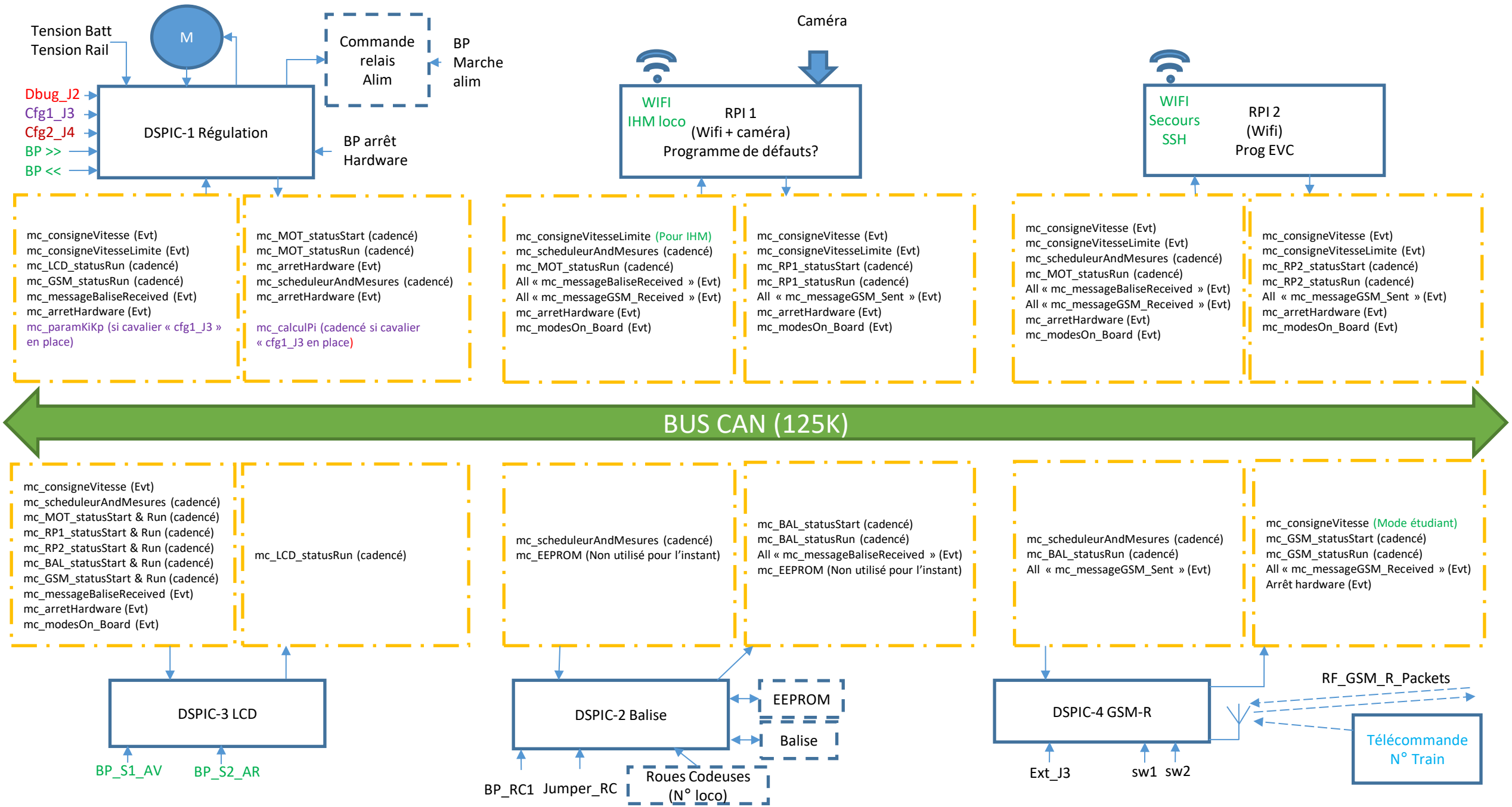
# Synoptique Train Trame Can

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

# Historique des indices

| INDICE | DATE | OBJET | REDACTION | APPROBATION |
|---|---|---|---|---|
| 3,12 | 26/08/20 | Intégration Vars Alim Securité | Gilles | |
| 3,13 | 17/09/20 | Correction des mises en forme | Mario | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# V3.1

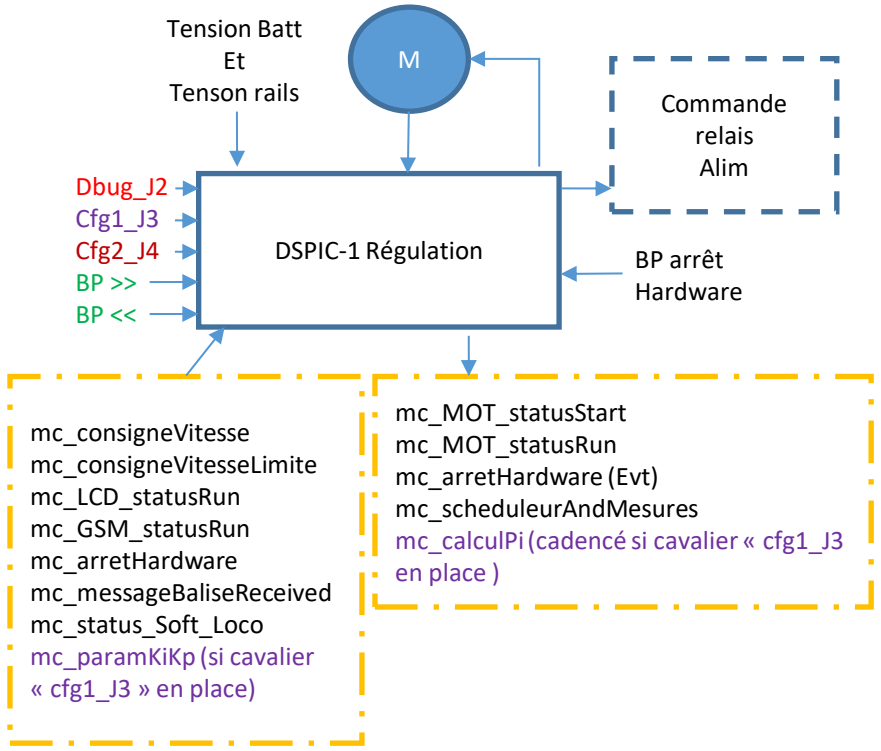

Tension Batt
Tension Rail
M
Commande relais Alim
BP Marche alim
Dbug_J2
Cfg1_J3
Cfg2_J4
BP >>
BP <<
DSPIC-1 Régulation
BP arrêt Hardware
Caméra
WIFI IHM loco
RPI 1 (Wifi + caméra) Programme de défauts?
WIFI Secours SSH
RPI 2 (Wifi) Prog EVC
mc_consigneVitesse (Evt)
mc_consigneVitesseLimite (Evt)
mc_LCD_statusRun (cadencé)
mc_GSM_statusRun (cadencé)
mc_messageBaliseReceived (Evt)
mc_arretHardware (Evt)
mc_paramKiKp (si cavalier « cfg1_J3 » en place)
mc_MOT_statusStart (cadencé)
mc_MOT_statusRun (cadencé)
mc_arretHardware (Evt)
mc_scheduleurAndMesures (cadencé)
mc_arretHardware (Evt)
mc_calculPi (cadencé si cavalier « cfg1_J3 en place)
mc_consigneVitesseLimite (Pour IHM)
mc_scheduleurAndMesures (cadencé)
mc_MOT_statusRun (cadencé)
All « mc_messageBaliseReceived » (Evt)
All « mc_messageGSM_Received » (Evt)
mc_arretHardware (Evt)
mc_modesOn_Board (Evt)
mc_consigneVitesse (Evt)
mc_consigneVitesseLimite (Evt)
mc_RP1_statusStart (cadencé)
mc_RP1_statusRun (cadencé)
All « mc_messageGSM_Sent » (Evt)
mc_arretHardware (Evt)
mc_modesOn_Board (Evt)
mc_consigneVitesse (Evt)
mc_consigneVitesseLimite (Evt)
mc_scheduleurAndMesures (cadencé)
mc_MOT_statusRun (cadencé)
All « mc_messageBaliseReceived » (Evt)
All « mc_messageGSM_Received » (Evt)
mc_arretHardware (Evt)
mc_modesOn_Board (Evt)
mc_consigneVitesse (Evt)
mc_consigneVitesseLimite (Evt)
mc_RP2_statusStart (cadencé)
mc_RP2_statusRun (cadencé)
All « mc_messageGSM_Sent » (Evt)
mc_arretHardware (Evt)
mc_modesOn_Board (Evt)
BUS CAN (125K)
mc_consigneVitesse (Evt)
mc_scheduleurAndMesures (cadencé)
mc_MOT_statusStart & Run (cadencé)
mc_RP1_statusStart & Run (cadencé)
mc_RP2_statusStart & Run (cadencé)
mc_BAL_statusStart & Run (cadencé)
mc_GSM_statusStart & Run (cadencé)
mc_messageBaliseReceived (Evt)
mc_arretHardware (Evt)
mc_modesOn_Board (Evt)
mc_LCD_statusRun (cadencé)
mc_scheduleurAndMesures (cadencé)
mc_EEPROM (Non utilisé pour l'instant)
mc_BAL_statusStart (cadencé)
mc_BAL_statusRun (cadencé)
All « mc_messageBaliseReceived » (Evt)
mc_EEPROM (Non utilisé pour l'instant)
mc_scheduleurAndMesures (cadencé)
mc_BAL_statusRun (cadencé)
All « mc_messageGSM_Sent » (Evt)
mc_consigneVitesse (Mode étudiant)
mc_GSM_statusStart (cadencé)
mc_GSM_statusRun (cadencé)
All « mc_messageGSM_Received » (Evt)
Arrêt hardware (Evt)
DSPIC-3 LCD
BP_S1_AV
BP_S2_AR
DSPIC-2 Balise
EEPROM
Balise
BP_RC1
Jumper_RC
Roues Codeuses (N° loco)
DSPIC-4 GSM-R
Ext_J3
sw1
sw2
RF_GSM_R_Packets
Télécommande N° Train

# Récapitulatif  des Identifiants Trame CAN

- Trames de « sécurité »
  - ConsigneLimiteUrgence
  - ArretHardware
- Trames « communication messages »
  - Messages Balises
  - Messages GSM_R
    - Train vers antenneTransfert trames CAN de la Carte Xbee vers carte Rpi du train.
    - Antenne vers train. Transfert trames CAN de la Carte Xbee vers carte Rpi du train.
- Trames des Status « Start » de tous les processeurs (uC ou Rpi) du train
- Trames des Status « Run » de tous les processeurs (uC ou Rpi) du train
- Trames gestion train
  - mc_scheduleurAndMesures
  - ConsigneVitesse
- Trames « Outils trains »
  - Lecture/Ecriture EEPROM
  - Param_Ki_Kp
  - Calcul_Pi

# Lexique codage trame CAN et variables associées

- MC : Message CAN pour constantes, mc: Message CAN pour variables.
- On se définit 6 diminutifs correspondant aux 6 processeurs d'une locomotive:
  - MOT: Carte régulation vitesse moteur. (Indice 0)
  - RP1 et RP2: Carte Rpi 1 et 2. (Indice 1 et 2)
  - BAL: Carte qui gère la communication avec deux balises. (Indice 3)
  - LCD: Carte affichage. (Indice 4)
  - GSM: Carte de communication Xbee. (Indice 5)
- Chaque variable identifiant un message CAN commencera par « mc » soit (message CAN), il sera suivi par un des 6 diminutifs ci-dessous si toutes ses datas sont gérées par cette même carte, ou si risque de confusion.
- Les datas de ces trames CAN peuvent être de différents formats: (bit, char, word, long ou float). De même que pour les noms des messages CAN, on peut insérer dans le nom des variables, un de ces 6 diminutifs si on a un risque de confusion.
- Pour les identifier, le nom de chaque variable sera codé de cette manière :
  - bdmc_NomVariable : « bit » data message CAN
  - cdmc_NomVariable : « char » data message CAN
  - wdmc_NomVariable : « word » data message CAN
  - ldmc_NomVariable : « long » data message CAN
  - float_NomVariable : « float » data message CAN




**mc_consigneVitesse**

IN

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 2 | xx | xx | | | | | | |

D[0] : cdmc_consigneVitesse
la vitesse en cm/s ( 0 <= consigne <= 50 ou 0x32) →Ecrêtage

D[1] :
bit0: bdmc_sensDeplacementLoco
le sens (bit0 = 1 -> sens Avant, bit0 = 0 -> sens Arrière)
bit1: bdmc_ForcageMvtLoco (Télécommande ou boutons « Wagon »)

NB : 16,944ms équivaut à 6,78mm à 40cm/s

**mc_consigneVitesseLimite**

IN

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 16 | 1 | xx | | | | | | | |

D[0] : cdmc_consigneVitesseLimite
la vitesse en cm/s ( 0 <= consigne <= 50 ou 0x32)

**Configuration cavaliers:**

Dbug_j2 = Sa présence permet l'affichage de données supplémentaires « type Debug » sur le LCD, ceci après l'appui sur le switch RC1 (var « bdmc_swRC1 » de mc_BAL_statusRun)

Cfg1_J3 → Accès aux paramètres d'asservissement (Ki, Kp, CalculPi).
Cfg2_J4 → Jumper « rouge » qui permet de « rattraper » le sens de rotation des moteurs.

**Fonction des poussoirs SW1 « << » et SW2 « >> »:**

Un appui sur l'un de ces poussoirs fera avancer ou reculer le train à une vitesse fixe. Un relâchement arrêtera le train.

**mc_MOT_statusStart**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | A0 | 1 | xx | | | | | | | |

D[0] : cdmc_MOT_versionSoft. (2x4bits version et release) ex 1,6 -> 0x16
PS: Cette trame est renvoyée jusqu'à ce que « bdmc_ACK_MOT_statusStart » de « mc_LCD_statusRun » soit « High ».

**mc_MOT_statusRun**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | 20 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Identification des erreurs survenues en cours de fonctionnement, Un niveau high quelque part provoquera l'arrêt du train en forçant sa vitesse à 0. Une intervention humaine sera nécessaire à la reprise.

bit0 : bdmc_MOT_crashSoft
bit1 : bdmc_MOT_securiteErreurPI (obstacle sur la voie ou erreur du PI trop élevée)
bit2 : Non réception de nouvelle consigne de vitesse (type homme mort) (Non codé)
bit3 : bdmc_stopLoco_XXX_CrashSoft (Fct OU de tous les bdmc_XXX_crashSoft)
bit4 : Reserved
bit5 : Reserved
bit6 : Reserved
bit7 : Reserved
PS: Union de tous ces bits : D[0]→ cdmc_MOT_erreurs

D[1] : Identification des « warnings » survenues en cours de fonctionnement, un niveau High quelque part indiquera une alerte. Le « Set et Reset » de ces bits est uniquement gérés par cette même carte.

bit0 : bdmc_MOT_initEnCours,
bit1 : bdmc_noPoweredByRails
bit2 : bdmc_batterieFaible
bit3 : Reserved
bit4 : bdmc_MOT_bugSoft
bit5 : bdmc_MOT_CAN_RxErrorPassive
bit6 : bdmc_MOT_CAN_TxErrorPassive
bit7 : bdmc_MOT_CAN_busOFF
PS: Union de tous ces bits : D[1]→ cdmc_MOT_warnings

D[2] : Input/output hardware Carte

bit0 : Reserved
bit1 : bdmc_MOT_jumper2 (Activation du mode Debug)
bit2 : bdmc_MOT_jumper3 (Jumper Cfg1)
bit3 : bdmc_MOT_jumper4 (Définit le sens de déplacement de la locomotive suivant le câblage des moteurs »
bit4 : bdmc_MOT_sw1 (Loco marche avant, Niveau High si appui)
bit5 : bdmc_MOT_sw2 (Loco marche arrière, Niveau High si appui)

NB: Les bits 1,2 et 3 ne sont lus qu'à la mise sous tension de la loco. Ils sont 'High' si positionnement du cavalier.

D[3] : Etat dynamique de la carte MOT

bit0 : Reserved
bit1 : bdmc_modeLocoOFF (Conf Trame mc_arretHardware)
bit2 : bdmc_modeMoveLocoManuelOrRemote
bit3 : bdmc_moveSensLocoManuelOrRemote
bit4 : Reserved
bit5 : Reserved
bit6 : Reserved
bit7 : Reserved
PS: Union de tous ces bits : D[3]→ cdmc_MOT_dynamique

D[4],D[5] : wdmc_tensionBatterie (D[5] :poids forts ,D[4] :poids faibles)

D[6] : cdmc_MOT_var1Debug
D[7] : cdmc_MOT_var2Debug

NB: Pour savoir si le déplacement de la loco est en mode « Remote control », il faut vérifier la variable « bdmc_modeRemoteControl » de la trame CAN « mc_GSM_statusRun ». Si oui, la vitesse et le sens seront aussi dans cette même trame.

NB: Avant,
D[4],D[5] : Tension rail sous la loco (D[5] :poids forts ,D[4] :poids faibles). NB: Pour des raisons de sécurité matérielle, une détection de tension trop forte sur les rails coupera instantanément le relais « énergie » de la loco.

**mc_scheduleurAndMesures** (Envoyée toutes les 16,944ms ce qui équivaut à 6,78mm à 40cm/s

**OUT**

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 2F | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : cdmc_ordonnancementId. Identifiant trames status (ordonnancement et synchronisation)
D[1] : cdmc_vitesseMesurée
D[2],D[3] : wdmc_QEI_distanceRelativeParcourue (PS: sera RAZ quand réception TrameBalise1)
D[4],D[5] : wdmc_valeurErreurRegul
D[6] : cdmc_vitesseConsigneInterne
D[7] : Reserved

NB QEI: Quadrature Encoder Interface (Périphérique uC)

**mc_paramKiKp**

**I/O Debug**

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 82 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

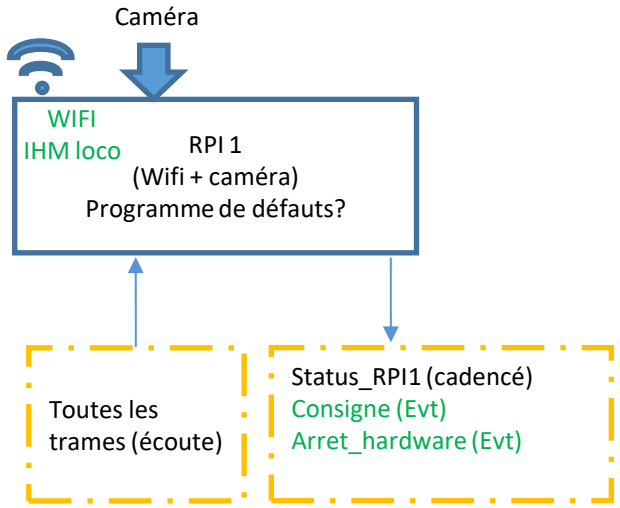
D[0],D[1],D[2],D[3] : fdmc_Ki →Valeurs du Ki en float
D[4],D[5],D[6],D[7] : fdmc_Kp →Valeurs du Kp en float

**mc_calculPi**

**OUT Debug**

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 83 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[3],D[2],D[1],D[0] : fdmc_erreurProportionnel.
D[7],D[6],D[5],D[4] : fdmc_erreurIntegrale




**mc_RP1_StatusStart**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | A1 | 5 | xx | xx | xx | xx | xx | | | |

D[0] : cdmc_RP1_versionSoft. (2x4bits version et release) ex 1,6 -> 0x16
D[1],D[2],D[3],D[4] : ldmc_adresseIP_RP1. (ex : 192.168.1.10)

PS: Cette trame est renvoyée jusqu'à ce que « bdmc_RP1_MOT_statusStart » de « mc_LCD_statusRun » soit « High ».

**mc_RP1_StatusRun**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | 21 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Erreurs: Un niveau high quelque part provoquera l'arrêt du train. Soit, vitesse = 0
- bit0 : bdmc_RP1_crashSoft
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved

PS: Union de tous ces bits : D[0]→ cdmc_RP1_erreurs

D[1] : Warnings
- bit0 : bdmc_RP1_initEnCours
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : bdmc_RP1_bugSoft
- bit5 : bdmc_RP1_CAN_RxErrorPassive
- bit6 : bdmc_RP1_CAN_TxErrorPassive
- bit7 : bdmc_RP1_CAN_busOFF

PS: Union de tous ces bits : D[0]→ cdmc_RP1_warnings

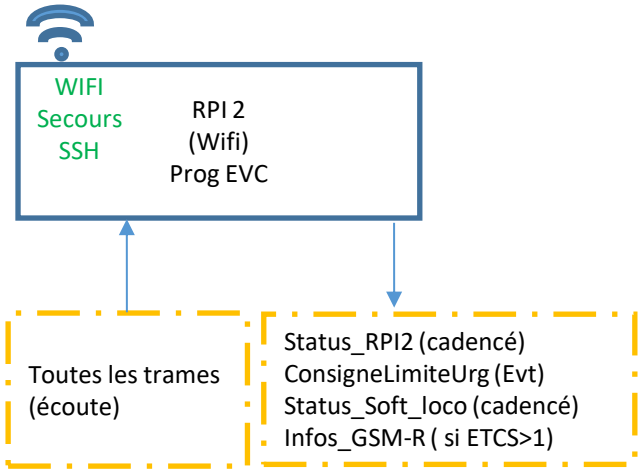
D[2] : Input/output hardware Carte
- bit0 : bdmc_presenceCamera (Low si absente)
- bit1 : bdmc_RP1_jumperCfg1 (High si cavalier présent)
- bit2 : bdmc_RP1_jumperCfg2 (High si cavalier présent)
- bit3 : Reserved
- bit4 : Reserved

D[3] : Etat dynamique de la carte
- bit0 : Reserved
- bit1 : bdmc_RP1_etatConnexion_TCP_IP (Low si absente)
- bit2 : bdmc_RP1_etatConnexionWIFI (Low si absente)
- bit3 : bdmc_etatConnexionCamera (Low si absente)
- bit4 : bdmc_modeRemoteControlMecanicien
- bit5 : bdmc_moveSensMecanicien (High → en avant, Low → en arrière
- bit6 : Reserved
- bit7 : bdmc_RP1_panicPowerOFF_loco

D[4] : cdmc_RP1_configONBOARD (en demande de validation)
- bits[3..0] : Mode « ON board » en demande de validation
- bits[5..4] : Niveau ERTMS en demande de validation

D[5] : Reserved

D[6] : cdmc_RP1_var1Debug
D[7] : cdmc_RP1_var2Debug




**Status_RP2_Start**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | A2 | 5 | xx | xx | xx | xx | xx | | | |

D[0] : cdmc_RP2_versionSoft (2x4bits version et release) ex 1,6 -> 0x16
D[1],D[2],D[3],D[4] : ldmc_adresseIP_RP2 (ex : 192.168.1.10)

PS: Cette trame est renvoyée jusqu'à ce que « bdmc_ACK_RP2_statusStart » de « mc_LCD_statusRun » soit « High ».

**Status_RP2_Run**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | 22 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Erreurs: Un niveau high quelque part provoquera l'arrêt du train. Soit, vitesse = 0
- bit0 : bdmc_RP2_crashSoft
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved

PS: Union de tous ces bits : D[0]→ cdmc_RP2_erreurs

D[1] : Warnings
- bit0 : bdmc_RP2_initEnCours
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : bdmc_RP2_bugSoft
- bit5 : bdmc_RP2_CAN_RxErrorPassive
- bit6 : bdmc_RP2_CAN_TxErrorPassive
- bit7 : bdmc_RP2_CAN_busOFF

PS: Union de tous ces bits : D[0]→ cdmc_RP2_warnings

D[2] : Input/output hardware Carte
- bit0 : Reserved
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved

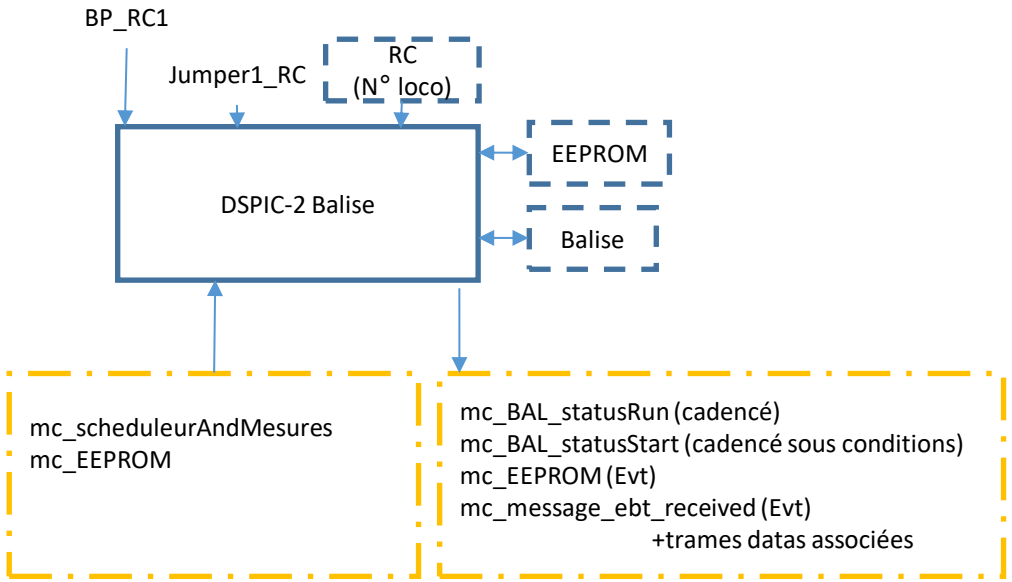
D[3] : Etat dynamique de la carte
- bit0 : Reserved
- bit1 : bdmc_RP2_etatConnexion_TCP_IP (Low si absente)
- bit2 : bdmc_RP2_etatConnexionWIFI (Low si absente)
- bit3 : Reserved
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : bdmc_RP2_panicPowerOFF_loco

PS: Union de tous ces bits : D[3]→ cdmc_RP2_dynamique

D[4] : Configuration « On Board » en cours de la locomotive
- bits[3..0] : Mode « ON board » en cours
- bits[5..4] : Niveau ERTMS en cours

D[5] : cdmc_vitesseModeMecanicien

D[6] : cdmc_RP2_var1Debug
D[7] : cdmc_RP2_var2Debug




**Fonctions de BP_RC1 et de Jumper1_RC:**

A la mise sous tension de la locomotive, le LCD est en mode affichage écran démarrage. C'est-à-dire que les versions logicielles des codes, les adresse TCP/IP, la tension batterie… sont affichées.
On restera dans ce mode jusqu'à ce que la loco reçoive une consigne de vitesse qui permette son déplacement.

Dès lors, deux cas possibles suivant la présence du cavalier « Dbug » sur la carte de gestion du moteur de la loco.

Cavalier absent: Mode normal affichage (dessin de la locomotive avec ses paramètres de vitesse et de déplacement, l'état du prochain feu sur la voie, les numéros de passage de balises, l'état de l'électrification de la voie ferrée….
Toutes ces données sont concentrées vers le bas de l'afficheur.

Cavalier présent: On utilisera en plus le haut du LCD pour afficher des variables de débogage. Ces variables ne sont pas destinées aux programmeurs « Utilisateur » de la maquette.

**mc_EbtL2_received**

I/O

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 0x30 | 8 | xx | xx | xx | xx | xx | xx | 00 | xx |

Pour coder le message Balise, on crée d'abord une trame CAN d'entête dont l'Id=0x30. Quant 'aux données du message, elles sont insérées dans d'autres trames CAN avec un identifiant incrémenté on insère les datas.
PS: Si la carte Balise envoie cette trame, alors les datas sont OK.

D[0] : cdmc_lenEbtL0

D[1] : cdmc_CKS_EbtL2Ext
est égal à somme des data[2] à data[7] pour l'Id = 0x30
additionné de la somme des Data[i] des « Id » supplémentaires et nécessaires.
PS: CKS_EbtL2Ext = CKS_EbtL2 + cdmc_CRO_comIR

D[2] : cdmc_standardCom (Protocole de communication)
D[3] : cdmc_dataSup
D[4] : cdmc_etatProchainFeu
D[5] : cdmc_sourceNumeroBalise
D[6] : 0x00, Reserved
D[7] : cdmc_CRO_comIR , Ce CRO indique les dysfonctionnements rencontres lors de la communication IR, mais au final, tout c'est bien passé.
[Bit3..Bit0] : représentent un compteur de dysfonctionnements
Bit4: Bad CKS
Bit5: Problème de TimeOUT dans la réception de toutes les datas

**mc_BAL_statustart**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | A3 | 1 | xx | | | | | | | |

D[0] : cdmc_BAL_versionSoft. (2x4bits version et release) ex 1,6 -> 0x16

PS: Cette trame est renvoyée jusqu'à ce que « bdmc_ACK_BAL_statusStart » de « mc_LCD_statusRun » soit « High ».

**mc_BAL_statusRun**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | 23 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Erreurs: Un niveau high quelque part provoquera l'arrêt du train. Soit, vitesse = 0
- bit0 : bdmc_BAL_crashSoft PS: En attente de codage
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : Reserved

PS: Union de tous ces bits : D[0]→ cdmc_BAL_erreurs

D[1] : Warnings
- bit0 : bdmc_BAL_initEnCours PS: En attente de codage
- bit1 : bdmc_BAL_BadComWithInfra Indique au moins une « Fail »Com IR
- bit2 : Reserved
- bit3 : Reserved
- bit4 : bdmc_BAL_bugSoft PS: En attente de codage
- bit5 : bdmc_BAL_CAN_RxErrorPassive
- bit6 : bdmc_BAL_CAN_TxErrorPassive
- bit7 : bdmc_BAL_CAN_busOFF PS: En attente de codage

PS: Union de tous ces bits : D[0]→ cdmc_BAL_warnings

D[2] : Input/output hardware Carte
- bit0 : bdmc_BAL_poids1RoueCodeuse
- bit1 : bdmc_BAL_poids2RoueCodeuse
- bit2 : bdmc_BAL_poids4RoueCodeuse
- bit3 : bdmc_BAL_poids8RoueCodeuse
- bit4 : bdmc_BAL_jumper1_RC
- bit5 : bdmc_BAL_swRC1
- bit6 : Reserved
- bit7 : Reserved

PS: Union de tous ces bits : D[2]→ cdmc_BAL_IO

D[3] : Etat dynamique de la carte
- bit0 : Reserved
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : bdmc_BAL_panicPowerOFF_loco

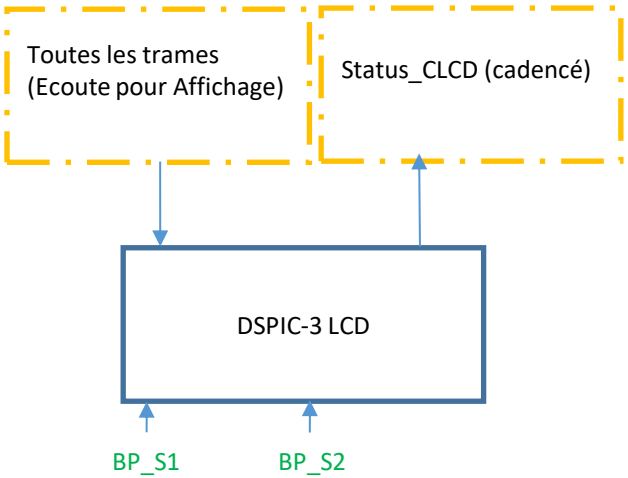
PS: Union de tous ces bits : D[3]→ cdmc_BAL_dynamique

D[4] : cdmc_BAL_statusComBalise
PS: ( déjà dans la 3eme trame balise)

D[5] : cdmc_BAL_lastNumberOfDetectedBali

D[6] : cdmc_BAL_var1Debug Compteur du nombre d'échecs de communication IR avec l'Infra
PS: Ce compteur est écrêté à 0xFF.

D[7] : cdmc_BAL_var2Debug Compte rendu (CRO) de la dernière com IR qui a échoué.




mc_LCD_statusRun

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| OUT | 24 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Erreurs:
- bit0 : bdmc_LCD_crashSoft
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : Reserved

D[0] : cdmc_LCD_erreurs

D[1] : Warnings
- bit0 : bdmc_LCD_initEnCours
- bit1 : Reserved
- bit2 : Reserved
- bit3 : En attente: Plantage LCD1/2 et procédure Reset LCD1/2
- bit4 : bdmc_LCD_bugSoft
- bit5 : bdmc_LCD_CAN_RxErrorPassive
- bit6 : bdmc_LCD_CAN_TxErrorPassive
- bit7 : bdmc_LCD_CAN_busOFF

D[1] : cdmc_LCD_warnings

D[2] : Input/output hardware Carte
- bit0 : bdmc_LCD_sw1 // Etat poussoir pour marche train avant
- bit1 : bdmc_LCD_sw2 // Etat poussoir pour marche train arrière
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved
- bit5 : Reserved

D[3] : Etat dynamique de la carte
- bit0 : Reserved
- bit1 : Reserved
- bit2 : bdmc_LCD_modeAccueil
- bit3 : bdmc_LCD_modeRun
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : Reserved

PS: Union de tous ces bits : D[3]→ cdmc_LCD_dynamique

D[4] : Acknowledge « statusStart ». Tous les processeurs autres que le processeur qui gère le « LCD » envoient au démarrage une trame de type mc_XXX_statusStart. Ceci pour signifier des paramétrages permanents comme par exemple leur version logicielle en place, leur adresse TCP... Tous ces paramètres sont affichés sur le LCD. Quand une réception est OK, on positionne un flag d'acquittement.
- bit0 : bdmc_ACK_MOT_statusStart (High quand ACK OK)
- bit1 : bdmc_ACK_RP1_statusStart (High quand ACK OK)
- bit2 : bdmc_ACK_RP2_statusStart (High quand ACK OK)
- bit3 : bdmc_ACK_BAL_statusStart (High quand ACK OK)
- bit4 : bdmc_ACK_GSM_statusStart (High quand ACK OK)

D[5] :
D[6] : cdmc_LCD_var1Debug
D[7] : cdmc_LCD_var2Debug

De façon contraire aux autres processeurs, aucune trame "mc_LCD_statusStart" n'a besoin de circuler sur le BUS. Le processeurs "LCD" connait sa version logicielle pour son affichage.

# Détail des différents écrans

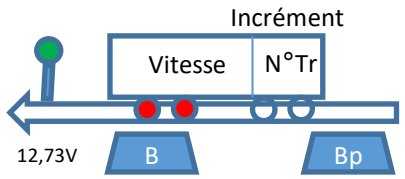
ECRAN démarrage

Train N°x
Soft MOT :
Soft BAL :
Soft GSM :
Soft LCD :
Soft RP1 :

192.168.123.100
TCP/IP:No
Soft RP2 :

192.168.123.100
TCP/IP:No
Batterie :12,73V

ECRAN Mode « Run »

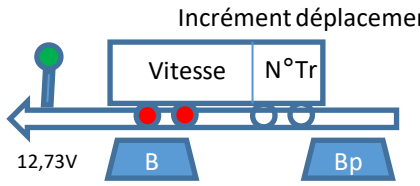



**Descriptions, décodage Affichage**
N°Tr: N° du train
Vitesse : Vitesse réelle locomotive
Incrément: Nombre impulsions codeurs (Odométrie)
PS: Remis à zero à chaque passage de balise
B: N° balise
Bp : N° balise précédente
12,73V : Tension batterie
Indication de la couleur du feu (vert, orange ou rouge)
Présence électrification rails : roues loco de couleur rouge

DSPIC-3 LCD

ECRAN Mode « Run » et « Debug »

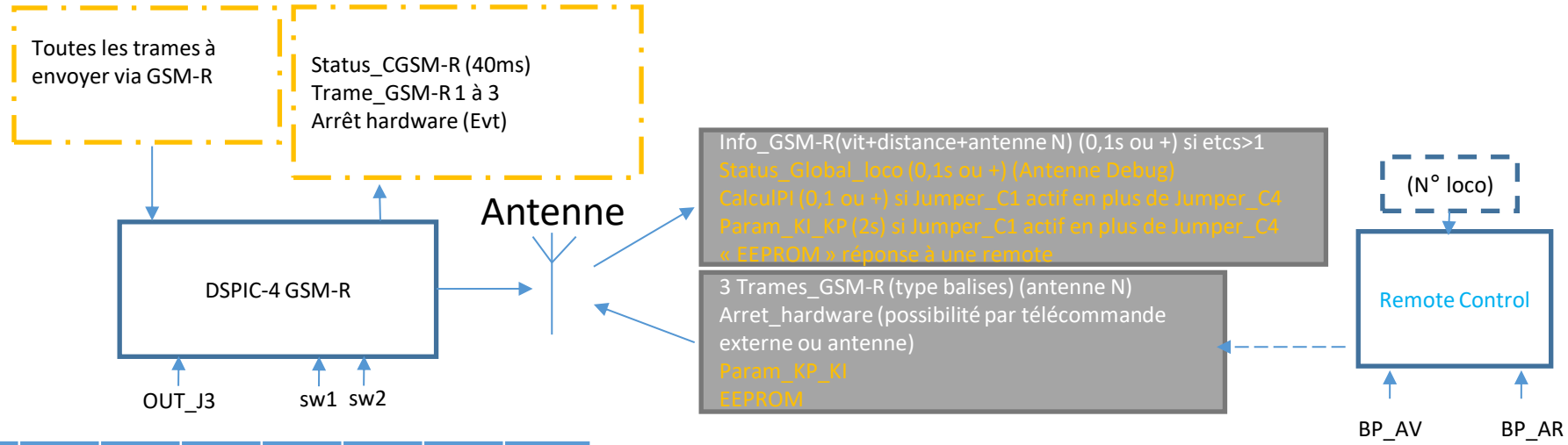
EE WW MOT DD x__ x__
EE WW RP1 DD x__ x__
EE WW RP2 DD x__ x__
EE WW BAL DD x__ x__
EE WW LCD DDx __ x__
EE WW GSM DD x__ x__




**Suppléments affichage du mode « Debug »**
EE : Affichage mode Hexa de Data[0], soit la data « erreur » de la trame CAN status du processeur
WW : Affichage mode Hexa de Data[1], soit la data « warning » de la trame CAN status du processeur
DD : Affichage mode Hexa de Data[3], soit la data « dynamique » de la trame CAN status du processeur

D[6] et D[7] sont les deux « Custom data » de la trame CAN status du processeur

Processeur: soit « MOT », « RP1 », « RP2 », « BAL », « LCD » et « GSM »




**mc_GSM_statusStart**

OUT

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| A5 | 1 | xx | | | | | | | |

D[0] : cdmc_GSM_versionSoft (2x4bits version et release) ex 1,6 -> 0x16
PS: Cette trame est renvoyée jusqu'à ce que « bdmc_ACK_GSM_statusStart » de « mc_LCD_statusRun » soit « High ».

**mc_GSM_statusRun**

OUT

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 25 | 8 | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : Erreurs: Un niveau high quelque part provoquera l'arrêt du train. Soit, vitesse = 0
- bit0 : bdmc_GSM_crashSoft //En attente codage
- bit1 : Reserved
- bit2 : Reserved
- bit3 : Reserved
- bit4 : Reserved

D[0] : cdmc_GSM_erreurs
D[1] : Warnings
- bit0 : bdmc_GSM_initEnCours //En attente codage
- bit1 : bdmc_GSM_failureACK_Antenne //En attente codage
- bit2 : Reserved
- bit3 : Reserved
- bit4 : bdmc_GSM_bugSoft //En attente codage
- bit5 : bdmc_GSM_CAN_RxErrorPassive
- bit6 : bdmc_GSM_CAN_TxErrorPassive
- bit7 : bdmc_GSM_CAN_busOFF

D[1] : cdmc_GSM_warnings

D[2] : Input/output hardware Carte
- bit0 : bdmc_GSM_switch1
- bit1 : bdmc_GSM_switch2
- bit2 : bdmc_GSM_switchOUT_J3
- bit3 : bdmc_GSM_Etat_Led_Rouge

D[3] : Etat dynamique de la carte
- bit0 : Reserved
- bit1 : Reserved
- bit2 : bdmc_GSM_modeRemoteControl
- bit3 : bdmc_GSM_moveSensRemoteControl (High → en avant, Low → en arrière)
- bit4 : Reserved
- bit5 : Reserved
- bit6 : Reserved
- bit7 : bdmc_GSM_panicPowerOFF_loco

PS: Union de tous ces bits : D[3]→ cdmc_GSM_dynamique

D[4] : cdmc_consigneVitesseRemoteControl . Valeur pour création de la consigne de vitesse en mode Remote Control.

D[5] : cdmc_GSM_Nb_FailRxPacketXbee

D[6] : cdmc_GSM_var1Debug : Variable du diagramme d'Etat « ETAT_RECEPTION_UART »
D[7] : cdmc_GSM_var2Debug

**mc_messageGSM_Sent**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| I/O | 0x60 | 8 | xx | xx | xx | xx | 00 | 00 | 00 | 00 |

Pour coder le message GSMR, on crée d'abord une trame CAN d'entête dont l'Id=0x60. Quant 'aux données du message, elles sont insérées dans d'autres trames CAN. Pour ce faire, on crée une trame CAN avec un identifiant incrémenté et on insère les datas (Max 8). On répète l'opération jusqu'à la fin des données du message GSMR. L'Id max est équivalent à 0x6C, en effet une longueur de message GSMR ne peut excéder 100 datas.

D[0] : cdmc_lenMessageSent
Longueur réelle du message GSMR, C'est aussi la somme des « DLC » de toutes les trames données du message à partir de l'Id (0x61).

D[1] : cdmc_customCKS_MessageSent
Somme des data[2] à data[7] pour l'Id = 0x60
additionné de la somme des Data[i] des « Id » supplémentaires et nécessaires. (Dépend de la longueur du message GSMR)

D[2],D[3] : wdmc_destAdressMessageSent
Adresse de destination de l'antenne Xbee.

D[4] : cdmc_frameId_NumMessage
MSB quartet == Numéro du train
LSB quartet == Numéro message GSM_R

D[5] : 0x00, Reserved
D[6] : 0x00, Reserved
D[7] : 0x00, Reserved

**mc_messageGSM_received**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| I/O | 0x70 | 8 | xx | xx | xx | xx | xx | 00 | 00 | 00 |

Pour coder le message GSMR, on crée d'abord une trame CAN d'entête dont l'Id=0x70. Quant 'aux données du message, elles sont insérées dans d'autres trames CAN. Pour ce faire, on crée une trame CAN avec un identifiant incrémenté et on insère les datas (Max 8). On répète l'opération jusqu'à la fin des données du message GSMR. L'Id max est équivalent à 0x7C, en effet une longueur de message GSMR ne peut excéder 100 datas.

D[0] : cdmc_lenMessageReceived
Longueur réelle du message GSMR, C'est aussi la somme des « DLC » de toutes les trames données du message à partir de l'Id (0x71).

D[1] : cdmc_customCKS_MessageReceived
est égal à somme des data[2] à data[7] pour l'Id = 0x70
additionné de la somme des Data[i] des « Id » supplémentaires et nécessaires. (Dépend de la longueur du message GSMR)

D[2],D[3] : wdmc_sourceAdressMessageReceived
Adresse de source de l'antenne Xbee.

D[4] : cdmc_RSSI_FromAntenne

D[5] : 0x00, Reserved
D[6] : 0x00, Reserved
D[7] : 0x00, Reserved

**mc_arretHardware**

| ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 1 | xx | | | | | | | |

Valeurs de D[0] : cdmc_causeLocoOFF

- 0x06 -> Notion d'arrêt d'urgence!!!!
- 0x01 → Appui sur bouton : BP arrêt hardware situé sur la loco.
- 0x02 → Tension de batterie trop faible.
- 0x03 → Détection d'inactivité de la loco prolongée → extinction automatique préservation batterie.
- 0x04 → Réception commande via télécommande externe ou une antenne.
  NB: La télécommande externe peut arrêter toutes les locos en même temps (Indice « 0 » pour sélection train)
- 0x05 → Réception commande d'arrêt via IHM RP1.
- 0x06 -> Détection collision avant du train (capteur sur Rpi avant de la loco)

La carte « MOT » peut émettre une trame « mc_arretHardware » dans les trois conditions suivantes: Appui BP « arret Hardware », Tension de batterie trop faible ou inactivité prolongée. Avant son envoi, la variable « cdmc_consigneVitesse » sera mise à 0x00 pour stopper la loco. Un compte à rebours interne sera aussi enclenché (environ 30s) et à son issu, le relais d'alimentation de la locomotive sera désactivée. (Extinction loco).

La variable « bdmc_modeLocoOFF » de la trame mc_MOT_statusRun devient High après l'envoi ou la réception d'une trame « mc_arretHardware ». Ceci indique aussi que le compte à rebours avant l'extinction de l'énergie de la loco est enclenché.

Actions à effectuer par les autres cartes processeurs:

-RP1 → Informer l'IHM et se mettre en « Mode Power Off »
-RP2 → Se mettre en « Mode Power Off »
-GSM → Informer Antenne ou RBC si mode ETCS>1 ????
-LCD → Afficher que la loco est en mode compte à rebours « Arrêt »
-BAL → Rien

Le RP1 peut aussi décider de l'extinction de la loco, de même que cet ordre peut aussi provenir via une télécommande externe. (Réception par la carte « GSM »). Pour ces deux cas, la carte RP1 ou GSM enverra une trame mc_arretHardware avec la data « cdmc_causeLocoOFF » à 0x04 ou 0x05.

Actions à effectuer par les autres cartes processeurs:

-MOT → Stopper la loco et enclencher le compte à rebours pour extinction Energie Loco.
-RP1 → Informer l'IHM et se mettre en « Mode Power Off »
-RP2 → Se mettre en « Mode Power Off »
-GSM → Informer Antenne ou RBC si mode ETCS>1 ????
-LCD → Afficher que la loco est en mode compte à rebours « Arrêt »
-BAL → Rien

**mc_EEPROM**

| | ID | DLC | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] |
|---|---|---|---|---|---|---|---|---|---|---|
| I/O | 1E | x | xx | xx | xx | xx | xx | xx | xx | xx |

D[0] : cdmc_commandeEEPROM
‘O’ (Code ASCII de O = 0x4F) → Paramétrage OFFSET
‘W’ (Code ASCII de W = 0x57) → Ecriture EEPROM
‘D’ (Code ASCII de D = 0x44) → Lecture EEPROM

D[1] : cdmc_data1EEPROM
D[2] : cdmc_data2EEPROM
ou
D[1],D[2] : wdmc_offsetEEPROM si D[0]=‘O’

D[3] : cdmc_data3EEPROM
D[4] : cdmc_data4EEPROM
D[5] : cdmc_data5EEPROM
D[6] : cdmc_data6EEPROM
D[7] : cdmc_data7EEPROM

NB: Pas d’auto-indentation de « wdmc_offsetEEPROM »

**Mode d’emploi EEPROM:**

- Paramétrage de l’adresse d’offset (pointeur):
  Le PC envoi une trame standard avec
  **Data[0]** = 'O' (Code ASCII de O = 0x4F)
  Suivi de la valeur de l'offset codé de la façon suivante:
  **Data[2]** pour poids forts et **Data[1]** pour poids faibles
  OFFSET = (unsigned int) ((Data[2] << 8 ) + Data[1] ) codage “Loco”
  **Data[1]** pour poids forts et **Data[2]** pour poids faibles
  OFFSET = (unsigned int) ((Data[1] << 8 ) + Data[2] ) codage “INFRA”
  (Soit inversion des poids forts et faibles !!!!!)

- Ecriture dans l’EEPROM :
  Le PC envoi une trame standard avec
  **Data[0]** = 'W' (Code ASCII de W = 0x57) suivi des valeurs à écrire dans l'EEPROM
  Le DLC indique le nombre de valeurs à écrire
- Lecture de l’EEPROM :
  Le PC envoi une trame de Remote avec l'Id EEPROM
  Le DLC de la trame remote indiquera le nombre de datas à renvoyer
  La carte renverra une trame avec
  **Data[0]** = 'D' (Code ASCII de D = 0x44) suivi du nombre de datas demandés.
  NB: Le DLC max de la trame de remote est de 7, car Data[0] = 'D',
  reste donc 7 octets de disponibles dans la trame de renvoi.