# Liste des ordres et format des données

Ici sont listés et décrits les ordres immédiats et longs. « ID » représente l'identifiant d'ordre.

Les ID de 0 à 127 sont réservés aux ordres binaires (ceux décrits dans ce document). Les ordres immédiats et longs peuvent utiliser les mêmes ID. Les ID de 128 à 255 sont réservés aux ordres ASCII, qui sont des ordres de réglage et de débogage.

Le champ « Data » contient une succession d'octets. Lorsque l'on veut écrire un nombre nécessitant plus d'un octet, on commence par les 8 bits de poids fort, et on continue ainsi jusqu'aux 8 bits de poids faible. De manière générale, on considère que le bit de poids fort est toujours transmis en premier. Il est possible d'avoir des champs dont la longueur n'est pas un multiple de 8, toutefois la taille totale du champ « data » en bits doit être un multiple de 8. Le format des données de ce champ varie d'un ordre à l'autre, et est explicité plus bas, en utilisant la syntaxe suivante :

`Nom_du_champ(taille_en_bits) [nom_val_1:valeur1, nom_val_2:valeur2]Nom_du_champ(taille_en_bits)`

*(Il y a dans cet exemple, deux nombres transmis, le deuxième prenant ses valeurs dans un sous ensemble valeurs des possibles sur 'taille_en_bits' bits).*

`Nom_du_champ` **:** explicite la signification des données contenues.
`Taille_en_bits` **:** les données sont écrites sur ce nombre de bits, à la suite du champ précédent *(même si le nombre de bits n'est pas un multiple de 8).*

Si pour un champ donné, les valeurs inscriptibles sur le nombre de bits donnés ne sont pas toutes des valeurs valides, il est possible de préciser explicitement les valeurs inscriptibles et leurs significations respectives.
`Nom_val_1` **:** Signification de la valeur #1
`Valeur1` **:** Valeur #1

Si un nombre est optionnel dans le champ « data », il sera représenté entouré d'accolades : `{}`
Il n'est possible de rendre un nombre optionnel que s'il est le dernier, ou bien si celui qui le succède est également optionnel.

Le symbole `{…}` indique que le champ précédent peut être répété autant de fois que voulu.

| VALUE_REQUEST | | | | VALUE_ANSWER |
|---|---|---|---|---|
| Nom | Description | ID | Data | Data |
| GetColor | Le bas niveau renvoie la couleur s'il la connait. | 0x59 | None | [BLEU:0x00, JAUNE:0x01, UNKNOWN:0x02]COLOR(8) |
| Ping | Ne fait rien. | 0X5A | None | None |
| AddTrajectoryPoints | Ajoute à la trajectoire courante un certain nombre de points (entre 0 et 31 points). | 0x5B | TRAJECTORY_INDEX(8) TRAJECTORY_POINT(56) {…} | None |
| SetMaxSpeed | Règle la vitesse maximale courante. | 0x5C | VITESSE_MAX_SIGNEE(16) | None |
| EditPosition | Modifie la position du robot dans le bas niveau, les coordonnées fournies sont ajoutées aux coordonnées courantes du bas niveau. | 0x5D | POSITION_XYO(40) | None |
| StopStream | Interromps le stream de position/capteurs (seul moyen de faire terminer l'ordre long "StreamAll") | 0x5E | None | None |
| SetSensorMode | Change de mode de rafraîchissement des capteurs. | 0x5F | SENSOR_MODE(8) | None |
| SetPosition | Modifie la position du robot dans le bas niveau, les coordonnées fournies deviennent les coordonnées courantes du bas niveau. | 0x60 | POSITION_XYO(40) | None |
| SetDirection | Règle la courbure courante des roues. | 0x61 | CURVATURE_BIS(16) | None |

| NEW_ORDER | | | | STATUS_UPDATE | EXECUTION_END |
|---|---|---|---|---|---|
| Nom | Description | ID | Data | Data | Data |
| FollowTrajectory | Suit la trajectoire courante. Se termine une fois arrivé au prochain point d'arrêt de la trajectoire, lorsque le robot est à l'arrêt. Ou bien si « Stop » est appelé. | 0x38 | VITESSE_MAX_SIGNEE(16) | None | [ARRIVED:0x00, EXT_BLOCKED:0x01, INT_BLOCKED:0x02, NO_MORE_POINTS:0x03, STOP_REQUIRED:0x04, FAR_AWAY:0x05]END_MOVE_STATUS(8) TRAJECTORY_INDEX(8) |
| Stop | Interromps la trajectoire courante au plus vite. La trajectoire est oubliée. Se termine quand le robot est à l'arrêt. | 0x39 | None | None | None |
| WaitForJumper | Se termine lorsque le jumper est retiré du robot. | 0x3A | None | None | None |
| StartMatchChrono | Se termine au bout de 90 secondes si tout se passe bien. Ou bien avant en cas de problème grave. | 0x3B | None | None | [MATCH_FINISHED:0x00, EMERGENCY_STOP :0x01]END_MATCH_STATUS(8) |
| StreamAll | Envoie la position du robot et/ou l'état des capteurs avec les fréquences demandées. | 0x3C | SEND_PERIOD(16) SENSORS_PRESCALER(8) | POSITION_XYO(40) TRAJECTORY_INDEX(8) {DIR_ANGLES(16) CAPTEURS(96)} | None |
| PullDownNet | Abaisse le filet à l'horizontale. | 0x3D | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| PutNetHalfway | Abaisse le filet à mi-chemin. Afin de pouvoir remplir un filet déjà en partie rempli sans perdre de balles. | 0x3E | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| PullUpNet | Remonte le filet à la verticale. | 0x3F | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| OpenNet | Ouvre les mailles du filet pour pouvoir accueillir les balles. | 0x40 | None | None | None |
| CloseNet | Ferme les mailles du filet. | 0x41 | None | None | None |
| CrossFlipFlop | Actionne le filet de manière à pouvoir traverser la bascule de la zone de départ en marche arrière comme si de rien était. Entre en CONFLIT avec tous les autres ordres actionnant le filet. | 0x42 | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| EjectLeftSide | Vide le filet par le côté gauche (du point de vue du robot). | 0x43 | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| RearmLeftSide | Range le bras gauche permettant de vider le filet. | 0x44 | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| EjectRightSide | Vide le filet par le côté droit (du point de vue du robot). | 0x45 | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |
| RearmRightSide | Range le bras droit permettant de vider le filet. | 0x46 | None | None | [SUCCESS:0x00, FAILURE:0x01]END_STATUS(8) |

| | | | | | |
|---|---|---|---|---|---|
| FunnyAction | Lance le projectile. Dure 5 secondes maximum. | 0x47 | None | None | None |
| LockNet | Serre les mailles du filet. | 0x48 | None | None | None |
| Scann | Fait tourner les roues doucement, vers la gauche puis vers la droite. Pour balayer l'environnement avec les capteurs. | 0x49 | None | None | None |
| CloseNetForce | Ferme les mailles du filet en forçant un peu. | 0x4A | None | None | None |
| MoveUranus | Règle la consigne en vitesse à maxMovingSpeed (en désactivant l'asservissement en position et sur trajectoire). | 0x4B | None | None | None |

## Format des variables

**TRAJECTORY_POINT** – [POSITION_XYO(40)][IS_STOP_POINT(1)][CURVATURE(15)]

**IS_STOP_POINT** – type : boolean

*Indique si le point en question de la trajectoire courbe est un point d'arrêt ou non. Si c'est un point d'arrêt le robot fera en sorte d'avoir une vitesse nulle au moment d'arriver sur ce point, et terminera l'ordre « FollowTrajectory » en cours.*

**CURVATURE** – type : int15_t – unit : $hm^{-1}$

*Courbure de la trajectoire, signée. Le bit de poids fort représente le signe (un 1 signifie « négatif »). Une courbure positive correspond à un virage à gauche, et inversement. L'unité utilisée est « l'inverse de l'hectomètre ». Ainsi le rayon de courbure minimal pouvant être écrit vaut environ 6mm, et la résolution dépasse celle de la mécanique.*

**CURVATURE_BIS** – type : int16_t – unit : $hm^{-1}$

*Représente une courbure destinée à régler l'orientation des roues. Avec les mêmes conventions que précédemment, mais cette fois le signe est codé de manière conventionnelle.*

**CAPTEURS** –
[CAPTEUR_LONG_RANGE(8)][CAPTEUR_IR(8)][CAPTEUR_LONG_RANGE(8)][CAPTEUR_IR(8)][CAPTEUR_SHORT_RANGE(8)]*8

*Les longueurs mesurées pas les différents capteurs sont transmises dans l'ordre suivant (ToF = Time of Flight ; IR = InfraRouge ; LP = Longue Portée), les plus attentifs remarqueront que l'ordre choisi s'inspire d'un cercle trigonométrique :*
*[1] ToF LP Avant ; [2] IR Avant Gauche ; [3] ToF LP Arrière ; [4] IR Avant Droit ; [5] ToF Avant Gauche ; [6] ToF Flan Avant Gauche ; [7] ToF Flan Arrière Gauche ; [8] ToF Arrière Gauche ; [9] ToF Arrière Droit ; [10] ToF Flan Arrière Droit ; [11] ToF Flan Avant Droit ; [12] ToF Avant Droit*

**CAPTEUR_SHORT_RANGE** – type : uint8_t – unit : mm

**CAPTEUR_LONG_RANGE** – type : uint8_t – unit : double-millimètre (donc il faut donc multiplier la valeur par deux pour obtenir la distance en millimètres)

**CAPTEUR_IR** – type : uint8_t – unit : cm

**POSITION_XYO** – [POSITION_XY(24)][ANGLE(16)]

**POSITION_XY** - [X(12)][Y(12)]

*Représente la position d'un point sur la table de jeu, en mm. Le système de coordonnées standard d'INTech translaté de -1500mm selon l'axe X est utilisé. Et on translate également selon l'axe Y de -1000mm, ainsi (0,0) représente le point (-1500, -1000) du repère INTech.*

**X** - type : uint12_t - unit : mm

**Y** - type : uint12_t - unit : mm

**VITESSE_MAX_SIGNEE** - type : int16_t - unit : mm/s

*Représente la vitesse maximale de translation que le robot est en droit d'atteindre lors de l'exécution d'une trajectoire. C'est également la vitesse qu'il tente d'approcher du mieux possible. C'est une grandeur algébrique, une vitesse négative correspond à une marche arrière.*

**END_MOVE_STATUS** - type : uint8_t

*Indique de résultat de l'exécution d'une trajectoire. Le résultat pouvant être : « robot bien arrivé à destination » (0x00)*
*Ou bien l'un des cas d'erreur suivants : « robot bloqué par un obstacle extérieur » (0x01), « roues du robot bloquées » (0x02), « il n'y a plus aucun nouveau point dans la trajectoire courante, et le dernier n'était pas un point d'arrêt » (0x03), « la commande 'stop' a été appelée » (0x04), « le robot se trouve trop loin de la position indiquée par le point de trajectoire courant » (0x05).*
*En cas d'erreur, le robot est immobilisé et la trajectoire courante est effacée.*

**END_MATCH_STATUS** - type : uint8_t

*Indique le résultat de l'exécution du match. Cela peut être : « match terminé au bout de 90 secondes comme prévu » (0x00) out bien « match terminé prématurément à cause d'un problème grave » (0x01). Concrètement, le seul 'problème grave' détectable par le bas niveau est le niveau critique de batterie. Mais ceci peut être utilisé pour signaler n'importe quel problème empêchant de terminer le match, et il est inutile de préciser explicitement le problème au haut niveau, de toute façon il ne pourra rien y faire.*

**SEND_PERIOD** - type : uint16_t - unit : ms

*Délai entre l'émission de deux trames consécutives. Si la valeur est trop faible, le délai réel sera le délai minimal réalisable compte tenu de l'occupation et de la vitesse de la liaison série.*

**SENSORS_PRESCALER** - type : uint8_t - unit : None

*Permet de ne pas envoyer les données des capteurs à chaque trame, toute les SEND_ PERIOD millisecondes, mais seulement dans une trame sur SENSOR_ PRESCALER. Si la valeur est 0, les données des capteurs ne sont jamais envoyées.*

**ANGLE** – type : uint16_t – unit : milli-radians
*Angle dans le repère standard INTech. Entre 0 et 2PI.*

**TRAJECTORY_INDEX** – type : uint8_t
*Les points d'une trajectoire sont effectués dans l'ordre de leur TRAJECTORY_INDEX. Il s'agit d'un indice sur un tableau circulaire (l'indice suivant 255 est 0).*

**COLOR** – type : uint8_t
*Indique soit le côté de la table où se situe le robot au début du match, soit que cette information n'est pas disponible.*

**DIR_ANGLES** – type : [uint8_t][uint8_t]
*Deux angles représentés en degrés. Le premier est l'angle de la roue gauche, le second est celui de la roue droite. Les angles sont toujours compris entre 0 et 255. L'origine se trouve à 150°, si les deux roues se trouvent à l'origine la trajectoire est rectiligne. Si les deux angles sont inférieurs à 150°, alors la trajectoire et un virage à gauche. A l'inverse deux angles supérieurs à 150° correspondent à un virage à droite.*

**SENSOR_MODE** – type : uint8_t
*Permet de sélectionner le mode de rafraîchissement des capteurs. Les différents modes sont les suivants :*

- *(0x00) NONE : aucun capteur n'est mis à jour*
- *(0x01) FRONT_AND_BACK : les 5 capteurs avant et les 3 capteurs arrière sont mis à jour. Fréquence : 28Hz*
- *(0x02) FRONT_AND_SIDES : les 5 capteurs avant et les 4 capteurs latéraux sont mis à jour. Fréquence : 25Hz*
- *(0x03) BACK_AND_SIDES : Les 3 capteurs arrière et les 4 capteurs latéraux sont mis à jour. Fréquence : 32Hz*
- *(0x04) ALL : Tous les capteurs (les 12) sont mis à jour. Fréquence : 19Hz*