Protocole d'automation du Trinnov Amethyst

Version 1.7

Rémy Bruno

19 mai 2015

1 Introduction

Le Trinnov Amethyst peut être piloté à distance par protocole TCP/IP ou par liaison série RS232. Ces deux modes de communication fonctionnent de façon très similaire par une liaison bidirectionnelle dans laquelle le client envoie des *commandes* et l'Amethyst répond à ces commandes et envoie des *messages* fournissant son état courant.

L'utilisation recommandée, quel que soit le protocole utilisé, est de se connecter à l'l'Amethyst, éventuellement de lui demander son état courant (cf. § 5), puis, alternativement, de lui envoyer des commandes et de lire les données qu'il envoie. Comme l'Amethyst envoie un *message* à chaque changement d'état (volume courant, profile et presets chargés, etc.), le client peut ainsi connaître à tout instant l'état de l'l'Amethyst.

2 Description générale du protocole

La communication s'effectue en utilisant des caractères ASCII. Chaque communication (*commande* ou *message*) s'effectue sous la forme d'une ligne de texte terminée par un caractère *retour chariot*.

Chaque ligne commence par un mot-clef constitué de caractères alphanumériques indiquant la commande ou le type de message, suivie éventuellement par des arguments séparés par des espaces. L'Amethyst reconnaît les trois types suivants de *retours chariot* : le caractère `\n` (ou `0x0A`), le caractère `\r` (ou `0x0D`), et les deux caractères à la suite `\r\n` (ou `0x0D 0x0A`). Ceci permet la compatibilité avec les systèmes de type Unix, Mac et Windows. Les *commandes* et leurs arguments sont sensibles à la casse des caractères (*case sensitive* en anglais), c'est-à-dire qu'il faut respecter les majuscules et les minuscules (`command` et `COMMAND` sont ainsi deux commandes différentes).

Chaque *commande* est suivie par un *message* de réponse du processeur qui est soit `OK` si la commande réussit, soit `ERROR` suivi d'une description de l'erreur survenue. Le processeur peut également envoyer d'autres *messages*, indiquant notamment un changement d'état, selon les effets de la *commande*. Il faut noter que la réponse n'intervient pas nécessairement avant les éventuels autres messages de l'l'Amethyst.

3 Protocole TCP/IP

Ce mode de communication utilise les connexions réseau TCP/IP disponibles sur l'Amethyst.

3.1 Configuration réseau

Le manuel utilisateur indique comment configurer le réseau de manière détaillée, ainsi, nous ne rentrerons pas dans les détails ici.

La configuration réseau RJ45 de l'l'Amethyst s'effectue par DHCP au moment du boot. Lors de sa requête DHCP, le client DHCP de l'l'Amethyst envoie un identifiant (option `dhcp-client-identifier` de *dhclient*) qui est `TRINNOV-SRP-<id>` où `<id>` est l'identifiant de la machine. Ceci permet par exemple de lui affecter une adresse IP spécifique automatiquement à l'aide d'une configuration appropriée du serveur DHCP.

Il est également possible de spécifier les paramètres réseau à la main sous l'onglet *Network* (ou *System Status* selon les versions) de la page *Setup* de l'l'Amethyst.

3.2 Paramètres TCP/IP du protocole

La communication s'effectue par protocole TCP sur le port 44100.

Ce protocole est ainsi tout à fait similaire au SMTP, FTP, ou même HTTP. Une première étape de prise en main de ce protocole peut ainsi consister à se connecter en utilisant un logiciel de type `telnet` pour taper les lignes de commandes au clavier.

3.3 Connexion

Lorsqu'on se connecte en TCP sur le port 44100, l'Amethyst envoie un message de bienvenue, spécifiant également la version de l'l'Amethyst ainsi que son identifiant unique. Cet identifiant (`ID` ou `SRPID`) est un nombre qui permet d'obtenir le numéro de série de la machine (donné par les 20

bits inférieurs de cet identifiant), ainsi que le type de machine (donné par les bits supérieurs). Le format de cette ligne est le suivant :

```
Welcome on Trinnov Optimizer (Version 3.8.7, ID 8388609)
```

Cet identifiant 8388609 s'écrit `0x800001` en hexadécimal, soit un type 8 de machine (correspondant au type Amethyst) et un numéro de série de 1.

L'Amethyst attend ensuite que le client s'identifie. Ceci est effectué en envoyant la commande `id` avec comme argument un identifiant propre au client. Cette identification n'est pas une authentification, et permet simplement un éventuel comportement spécifique de l'l'Amethyst avec certains clients. Par exemple :

```
id mon_système_domotique
```

La communication s'effectue ensuite selon le mode de *commandes* et de *messages* présenté dans la section 2 et détaillé dans la section 5. Il faut noter que le traitement des commandes ne s'effectue qu'après que le client s'est identifié par la commande `id`.

4 Protocole RS232

L'Amethyst peut également être contrôlé par liaison RS232 à l'aide d'un câble *null modem* à 9 broches. Les paramètres du port série RS232 sont les suivants :

- 19200 bauds
- 1 stop bit
- 8 data bits
- pas de bit de parité

Dès qu'un client se connecte sur le port série, l'Amethyst est dans le mode de *commandes* et de *messages* présenté dans la section 2 et détaillé dans la section 5. Il ne faut donc pas s'identifier, contrairement au protocole TCP/IP.

Veuillez noter que toutes les machines ne sont pas équipées d'un connecteur RS232. Nous vous invitons à vérifier les connecteurs disponibles sur votre machine.

5 Commandes

Voici une liste des commandes reconnues. Cette liste n'est pas exhaustive mais inclut les commandes servant pour une utilisation normale de l'l'Amethyst. Les arguments de chaque commande sont symbolisés entre <...>.

- `volume` <*volume*>
  Cette commande permet de changer le volume principal à la valeur <*volume*> dB. Il s'agit du *master level* apparaissant dans l'interface graphique de l'l'Amethyst.
- `dvolume` <*delta*>
  Cette commande permet d'ajouter au volume principal la valeur <*delta*> dB. Cette valeur peut être positive ou négative.
- `volume_ramp` <*cible*> <*durée*>
  Cette commande permet d'appliquer une rampe de volume qui part du volume courant pour arriver au volume <*cible*> (en dB) après <*durée*> milisecondes. Cette commande est disponible à partir de la version 3.7.3 de l'l'Amethyst.
- `mute` <*action*>
  Cette commande permet de modifier l'état du *mute* de l'l'Amethyst :
  - si <*action*> vaut 0, le *mute* est désactivé (le son est réactivé) ;
  - si <*action*> vaut 1, le *mute* est activé (le son est désactivé) ;
  - si <*action*> vaut 2, l'état du *mute* est inversé.
- `dim` <*action*>
  Cette commande permet de modifier l'état du *dim* de l'l'Amethyst :
  - si <*action*> vaut 0, le *dim* est désactivé ;
  - si <*action*> vaut 1, le *dim* est activé ;

  - si <*action*> vaut 2, l'état du *dim* est inversé.
- `bypass` <*action*>
  Cette commande permet de modifier l'état de *bypass* de l'l'Amethyst :
  - si <*action*> vaut 0, le *bypass* est désactivé ;
  - si <*action*> vaut 1, le *bypass* est activé ;
  - si <*action*> vaut 2, l'état du *bypass* est inversé.
- `send_volume`
  Cette commande demande à l'l'Amethyst de renvoyer des messages donnant les informations d'état courant liées au volume, en particulier les messages `VOLUME`, `MUTE`, `DIM` et `BYPASS` (cf ces messages dans le paragraphe 6). Cette commande est disponible à partir de la version 3.5 de l'l'Amethyst.
- `remapping_mode` <*mode*>
  Cette commande permet de modifier le mode de remapping :
  - si <*mode*> vaut `none`, le remapping est désactivé ;
  - si <*mode*> vaut `2D`, le remapping 2D est activé ;
  - si <*mode*> vaut `3D`, le remapping 3D est activé ;
  - si <*mode*> vaut `autoroute`, le mode autorouting est activé ;
  - si <*mode*> vaut `manual`, le remapping manuel est activé.
- `use_acoustics_correction` <*action*>
  Cette commande permet de modifier l'état de la correction automatique :
  - si <*action*> vaut 0, la correction acoustique est désactivée ;
  - si <*action*> vaut 1, la correction acoustique est activée ;
  - si <*action*> vaut 2, l'état de la correction acoustique est inversé.
- `use_level_alignment` <*action*>
  Cette commande permet de modifier l'état de l'alignement de niveau :
  - si <*action*> vaut 0, l'alignement de niveau est désactivé ;
  - si <*action*> vaut 1, l'alignement de niveau est activé ;
  - si <*action*> vaut 2, l'état de l'alignement de niveau est inversé.
- `use_time_alignment` <*action*>
  Cette commande permet de modifier l'état de l'alignement temporel :
  - si <*action*> vaut 0, l'alignement temporel est désactivé ;
  - si <*action*> vaut 1, l'alignement temporel est activé ;
  - si <*action*> vaut 2, l'état de l'alignement temporel est inversé.
- `quick_optimized` <*action*>
  Cette commande permet de modifier l'état de l'optimisation :
  - si <*action*> vaut 0, l'optimisation est désactivée ;
  - si <*action*> vaut 1, l'optimisation est activée ;
  - si <*action*> vaut 2, l'état de l'optimisation est inversé.
- `change_page` <*delta*>
  Cette commande change la page actuellement affichée à l'interface. La valeur de <*delta*> indique de combien de pages il faut changer, et peut être positive ou négative (1 pour descendre d'une page, -1 pour remonter d'une page).
- `loadp` <*preset*|*fichier*>
  Cette commande permet de charger le preset numéro <*preset*>. Les presets utilisateur commencent au numéro 1, le preset 0 correspond au built-in (preset d'usine). Cette syntaxe est disponible depuis la version 3.3.4 de l'l'Amethyst.
  Pour les versions plus anciennes, il faut utiliser la syntaxe <*fichier*>, décrite ci-dessous, où l'on fournit un nom de fichier au lieu d'un numéro de preset. Par compatibilité avec les versions antérieures à la 3.3.4, les versions plus récentes supportent également cette syntaxe <*fichier*>. L'argument <*fichier*> est le nom d'un fichier qui dépend du numéro de preset à charger et est formé ainsi :

    `Config_`n`.xml`

où *n* est à remplacer par le numéro de preset. Par exemple, pour charger le preset numéro 1, il suffit d'envoyer la ligne de commande suivante :

```
loadp Config_1.xml
```

Les lignes de commandes étant *case sensitive*, il faut bien mettre une majuscule au nom du fichier du preset `Config_1.xml`. Note : cette deuxième syntaxe ne permet pas de charger le preset built-in.

- `get_current_preset`
  Cette commande renvoie le numéro du preset courant. Cette commande est disponible à partir de la version 3.3.4 de l'l'Amethyst.
- `get_label` <*n*>
  Cette commande renvoie le nom du preset numéro <*n*>. Cette commande est disponible à partir de la version 3.3.4 de l'l'Amethyst.
- `get_all_label`
  Cette commande renvoie la liste de tous les presets présents sur la machine, avec leurs numéros et leurs noms. Cette commande est disponible à partir de la version 3.3.4 de l'l'Amethyst.
- `profile` <*source*>
  Cette commande permet de changer de source (également appelé profile). L'argument est le numéro de source, où la première source porte le numéro 0.
- `tac_preset` <*preset*>
  Cette commande est un ancien alias pour la commande `profile`.
- `get_current_profile`
  Cette commande renvoie le numéro de la source (également appelé profile) courante. Cette commande est disponible à partir de la version 3.5.0rc8 de l'l'Amethyst.
- `get_profile_name` <*n*>
  Cette commande renvoie le nom de la source (ou profile) numéro <*n*>. Il s'agit du nom tel qu'il apparaît dans la page de sélection de source de l'interface graphique. Cette commande est disponible à partir de la version 3.5.0rc8 de l'l'Amethyst.
- `bye`
  Cette commande demande au serveur de clore la connexion. Le serveur répond alors en envoyant `BYE` (au lieu de `OK`) et en fermant la connexion. Les commandes `exit` et `quit` sont des synonymes. Cette commande n'a pas d'effet sur le fonctionnement de l'l'Amethyst (pas d'extinction ni de disparition du son), mais uniquement sur la connexion client-serveur. Cette commande ne doit pas être utilisée avec le protocole RS232.

6 Messages

Voici une liste, également non exhaustive, des principaux messages envoyés par l'Amethyst aux clients qui lui sont connectés :

- `VOLUME` <*vol*>
  Ce message indique le volume principal courant (en dB).
- `DIM` <*dim*>
  Ce message indique si le dim est actif (<*dim*> à 1) ou pas.
- `MUTE` <*mute*>
  Ce message indique si le mute est actif (<*mute*> à 1) ou pas.
- `BYPASS` <*bypass*>
  Ce message indique si le bypass est actif (<*bypass*> à 1) ou pas.
- `META_PRESET_LOADED` <*profile*>
  Ce message indique le profile (ou source) actif.
- `SRATE` <*srate*>
  Ce message donne la fréquence d'échantillonnage courante de l'l'Amethyst.
- `AUDIOSYNC_STATUS` <*status*>
  Ce message indique si l'Amethyst est correctement synchronisé sur la source audio (<*status*> à 1) ou pas.
- `AUDIOSYNC` <*mode*>
  Ce message indique le mode de synchronisation de l'l'Amethyst, qui peut être *Master* ou *Slave*.

– `SPEAKER_INFO` <*numéro_hp*> <r> <θ> <ϕ>
  Ce message est envoyé pour chaque haut-parleur calibré quand un preset est chargé. L'argument <*numéro_hp*> indique à quel haut-parleur correspondent les informations qui suivent (les numéros de haut-parleurs commencent ici à 0). Les arguments <r>, <θ> et <ϕ> donnent la position du haut-parleur correspondant en coordonnées sphériques (r en mètres, θ en degrés par rapport au plafond (pôle nord), ϕ en degrés par rapport à l'avant et positif vers la gauche).
– `START_RUNNING`
  Ce message est envoyé lorsque l'Amethyst est prêt à faire du son. Ce message est par exemple reçu très peu de temps après l'envoi d'une commande `loadp` (typiquement, quelques dixièmes de secondes après).
– `LABELS_CLEAR`
  Ce message indique que la liste intégrale des presets présents sur la machine va être envoyée par l'Amethyst juste après ce message, et donc que le client, s'il maintient une liste des presets disponibles, doit effacer cette liste avant de la remplir par les noms de presets qui vont suivre. Ce message est envoyé en particulier en réponse à la commande `get_all_label` juste avant la liste des presets.
– `LABEL` <n>: <*name*>
  Ce message indique que le preset numéro <n> existe et a pour nom <*name*>. Ce message est envoyé en particulier en réponse aux commandes `get_label` et `get_all_label`.

Il faut noter que les clients connectés doivent être prêts à recevoir à tout moment un message de la part de l'l'Amethyst. Lorsqu'un utilisateur modifie le volume courant, par exemple, quel que soit le moyen par lequel il le fait, l'Amethyst envoie à tous les clients connectés un message indiquant le nouveau volume courant. Également, lorsqu'un calibrage est lancé à l'aide de l'interface graphique, l'Amethyst envoie aux clients connectés les coordonnées des haut-parleurs, de même que lors de la commande `loadp`. Ceci permet à l'l'Amethyst d'interagir avec le client connecté. Il s'agit donc davantage d'un « dialogue » entre le client et le serveur que d'un protocole « ordre/réponse ». En d'autres termes, le flux est bidirectionnel en permanence, plutôt que monodirectionnel alternativement dans un sens ou dans l'autre.

A contrario, il est très vivement recommandé de ne pas faire de *polling*, c'est-à-dire lire régulièrement l'état de l'l'Amethyst, à plus forte raison si cela doit s'effectuer sous la forme de connexions et de déconnexions fréquentes à la machine. Puisque l'Amethyst envoie aux clients connectés tout changement d'état, il est inutile et délétère de demander constamment à la machine son état courant. L'approche recommandée consiste à se connecter une fois pour toutes à l'l'Amethyst, de lire les variables d'état pertinentes pour le client, et d'attendre les éventuels changements (volume, par exemple) envoyés par l'Amethyst. Cependant, des connexions et déconnexions occasionnelles à la machine par un client ne posent pas de problème.

7 Exemple

Voici un exemple de communication en TCP/IP entre l'Amethyst et un client. Le client utilisé est simplement le programme `telnet` que l'on trouve par exemple sur tout système Unix (dont Linux). La première ligne est la ligne de commande *shell* utilisée pour lancer le client, les lignes commençant par « > » sont les lignes entrées par l'utilisateur (sans le « > »), les autres lignes sont les réponses de l'l'Amethyst ou du client `telnet`.

Cet exemple est transposable quasiment à l'identique dans le cas du protocole RS232, en remplaçant `telnet` par un programme permettant de communiquer via le port série et en supprimant les trois premières lignes de la communication (à savoir le message *Welcome* de l'l'Amethyst, la commande *id*, ainsi que la réponse *OK* de l'l'Amethyst) et les deux dernières lignes de cette communication (la commande *bye* et la réponse *BYE*).

```
$ telnet srp 44100
Trying 192.168.77.10...
Connected to srp.
Escape character is '^]'.
Welcome on Trinnov Optimizer (Version 3.8.7, ID 123456)
>id Système domotique de Trinnov Audio
OK
>loadp config_1.xml
```

```
ERROR: non-existing or inaccessible config file
>loadp Config_1.xml
OK
SPEAKER_INFO 0 1.36485 102.091 -43.3817
SPEAKER_INFO 1 1.21479 100.781 28.5073
START_RUNNING
>volume -12
OK
VOLUME -12.000000
>dvolume 1
OK
VOLUME -11.000000
>mute 2
OK
MUTE 1
>mute 2
OK
MUTE 0
>volume_ramp -20 50
OK
VOLUME -11.000000
VOLUME -11.960000
VOLUME -12.920000
VOLUME -13.880000
VOLUME -14.840000
VOLUME -15.800000
VOLUME -16.760000
VOLUME -17.720000
VOLUME -18.680000
VOLUME -20.000000
>bye
BYE
Connection closed by foreign host.
```

8 Conclusion

Ce protocole de communication est sujet à évolutions, mais, le cas échéant, ces évolutions iront toujours dans le sens de rester compatibles avec le fonctionnement antérieur. Cette approche de compatibilité permet d'expliquer que certains noms de commandes puissent sembler incohérents d'un point de vue de leur nomenclature (par exemple `tac_preset` et `get_current_profile`).

Les questions sont bienvenues, et doivent être adressées à `remy.bruno@trinnov.com`.