**Sommaire – PV Count\_Converter**

[PV du 18.01.19 + 21.01 - Concerne : RCO/YM 2](#_Toc11400876)

[PV du 22.02.19 2](#_Toc11400877)

[PV du 27-28.02.2019 2](#_Toc11400878)

[PV du 04.03.2019 3](#_Toc11400879)

[PV du 15.03.2019 3](#_Toc11400880)

[PV du 29.03.2019 3](#_Toc11400881)

[PV du 17.05.2019 3](#_Toc11400882)

[PV du 24.05.2019 3](#_Toc11400883)

# PV du 18.01.19 + 21.01 - Concerne : RCO/YM

Fait :

Revu du code logs et setenv\_c de l’oss fait pas YM

A faire :

Travailler par module :

* Main : CountConverter
* Setup
* Comptage
* Batch : setter les variables d’environnement se référer à son code setenv\_C

Le tout sur le N:\applications\Count\_Converter

Supprimer la partie GUI pour l’instant (pas de case à cocher)

* L’user sélectionnera les fichiers qu’il veut au début du lancement du programme

Commencer par travailler avec programme de conversion qui mettra les données dans un array puis mettra dans le fichier de sortie, créé dans un deuxième temps.

Pour l’instant ne pas travailler avec les fonctions dates et heure mais avec des entiers.

# PV du 22.02.19

L’entier du code de lecture et de récupération de données a été fait dans une seule fonction à la place de deux. Les données sont maintenant prêtes pour être convertie en format de sortie.

Il faut maintenant rajouter une nouvelle classe qui s’appelle Compteur\_1\_mois, qui fera appel au tableau « compteur[] » de 31x24, de la classe Compteur. Une fonction « add\_data\_M720 » qui aura pour paramètre une instance de la classe readM720. Dans cette fonction seront repris le no de canal, une variable noLigne aura pour valeur le (nombre de canal \* 2) puis une fonction qui bouclera sur les données du M720 pour les transformer en format de sortie.

# PV du 27-28.02.2019

Plusieurs classes ont été recrées. Les variables globales ont été supprimées et instanciées dans \_\_init\_\_() en tant que variable d’instance. Un tableau data\_int a été créé pour y insérer les valeurs de data\_ascii en int dedans. La question du traitement lors de plusieurs fichiers (comme par exemple 1 fichier par semaine donc 4-5 fichiers pour un mois à traiter) a été abordée et une modification devra être nécessaire en début du readFile().

Une fonction qui créera le tableau de sortie va être codée.

# PV du 04.03.2019

Faire deux algorithmes qui vont chercher les données pour un jour x et le positionner dans le tableau data\_int . A cet effet, une fonction de conversion est créée avec le code déjà présent dans le programme afin de pouvoir comparer les dates du fichier d’entrée (jjmmaa) et de sortie (aammjj)

Puis l’autre va définir comment l’utilisateur va fournir les fichiers à traiter. Soit :

* L’user donne un fichier et l’algorithme va définir à l’aide du nom du fichier les suivants qu’il faudra incorporer dans le traitement
* L’user donne tous les fichiers nécessaires

# PV du 15.03.2019

Suite à la création de la fonction ReadCan(can\_source), la lecture du fichier de config doit se faire selon l’exemple effectué par Yves.

# PV du 29.03.2019

Il a été fixé que l’utilisateur devra donner le fichier de début dont la suite sera calculée (via la suite d’hexadécimale). Les fichiers seront lus puis stockés dans un tableau avant de commencer la conversion.

L’organigramme fait par YM doit être mis au propre en structogramme.

# PV du 17.05.2019

YM a demandé à faire appel des variables/constantes Name et Version de Setup pour les utiliser dans le log. Puis avec le code trouvé sur les logs, il a été demandé de trouver une manière pour utiliser ce code de manière globale, afin de pouvoir les utiliser dans toutes les classes.

# PV du 24.05.2019

Le code trouvé est trop complexe pour la simple utilisation nécessaire. On a essayé de travailler avec mais la variable « nom\_fichier » qui permettait de display de quel module provient le log, était difficilement modifiable selon la classe où elle était utilisée. Je vais donc supprimer ces lignes pour remplacer avec le code d’Yves et faire appel à cette classe dans le readM720.

**Pv du 27.05.2019**

L’appel de la fonction MyLogging ne fait pas selon le code d’YM et donc que mettre une classe pour les logs n’étaient pas correctes. Pour ne pas se fatiguer à trouver une solution, j’ai proposé de faire une ligne de log pour chaque classe et ne pas faire appel à une instanciation. YM a demandé de faire une ligne de log sur la readM720 et on implémentera plus tard si nécessaire d’autres lignes dans les autres classes.

**PV du 28.05.2019**

Le travail de début de conversion a été discuté afin de faire un structo. Il y aura donc 2 boucles, une for nombre de canaux et une for nbre de jour puis le tout va recopier les données dans le tableau de sortie.

**PV du 14.06.2019**

Le structo a été revu et accepté. Un test concernant le nbre de canaux a été abordé mais comme ce test fait déjà l’objet de la boucle principale, il n’a pas lieu d’être en fin de boucle. La mise au propre et la codification du structo peut être amorcée.

**PV du 21.06.2019**

Suite au contrôle des structos, il manquait l’instanciation des variables en début des fonctions, les paramètres nécessaires à l’appel de la fonction secondaire ainsi que le renommage de certaines variables/fonction. Puis la seconde fonction était et incomplète. Nous l’avons refaite et j’ai remis au propre les 2 structos pour vérification.

**PV du 25.06.2019**

Le programme fonctionne et la conversion est opérationnelle. La mise en place d’un organigramme est à effectuer afin d’avoir une aperçu général du programme. Suite à la correction de ce dernier, je peux me mettre à la construction du main().