RSF
45, Avenue Marcel Dassault
Parc de la Plaine
31500 Toulouse
France
internet:www.rsf-europe.com
e-mail:rsf@rsf-europe.com

ProDVP MiniDVP
Le Script

Version documentation Provisoire

RSF

Les scripts

Ce document détaille les fonctionnalités offertes par les scripts, ainsi que la syntaxe des différentes instructions.

Introduction

Le DVP peut réaliser des scénarios complexes définis par le contenu de ce qu'on appelle des "fichiers scripts".

Les "fichiers scripts" sont des fichiers texte lisibles. On peut les modifier à l'aide de n'importe quel éditeur de texte.

Remarque préliminaire importante : la syntaxe est "sensible à l'utilisation des majuscules et des minuscules".

Déclarations

Les instructions de type "déclaration" définissent les types d'entités qui sont commandées ou contrôlées par les scripts

Clip

L'instruction "clip" vise à associer un nom logique (représenté par ClipName) à un fichier MPEG donné.

Syntaxe :

Clip ClipName("NomDeFichier");

ou Clip ClipName("NomDeFichier",t);

Dans le dernier cas, le paramètre "t" fixe la durée de l'exécution du clip. Si par exemple "t" = 20, alors la lecture du clip s'arrêtera après 20 secondes.

Exemples

Clip alive(“alive.MPG”);

Clip letstick(“letstick.MPG”);

Clip lord(“lord.VOB”,20);

Remarque : le “NomDeFichier” ne contient aucune indication sur l’emplacement effectif du fichier, mais seulement son nom et son extension.

PlayList (Liste de reproduction)

L’instruction “Playlist” vise à associer un nom logique (représenté par ListName) à une séquence de clips.

Syntaxe :

PlayList ListName(ClipName1, ClipName2, …, ClipNameN);

ou : PlayList ListName(NomDeFichier1,…, NomDeFichierN);

Exemples:

PlayList Hitparade(alive, letstick);

PlayList Hitparade(“alive.MPG”, ”letstick.MPG”);

PlayListFile (Fichier externe contenant une liste de reproduction)

L’instruction “PlayListFile” vise à associer un nom logique (représenté par ListName) à un fichier externe donné. Ce fichier texte externe, ayant l'extension ".pvp", inclura une liste de fichiers MPEG. L'usage de fichiers externes permet de les modifier pendant l'exécution d'un script, sans devoir l'arrêter.

Syntaxe :

PlayListFile ListName(NomDeFichier);

Exemples:

PlayList Hitparade ("musique.pvp");

Le contenu du fichier (de type texte) "musique.pvp" pourrait être :

alive.mpg

letstick.mpg
heyjude.mpg

Timer (Temporisateur)

L'instruction "Timer" vise à associer un nom logique (représenté par TimerName) à un temporisateur interne et à spécifier la durée, exprimée en seconde, au bout de laquelle la temporisation expire.

Syntaxe: Timer TimerName(durée);

Exemples:

Timer uneminute(60);

Timer dixsecondes(10);

Certaines entités qui sont traitées par les scripts sont implicitement déclarées (noms réservés). Ces entités sont :

Time (Heure)

L'instruction "Time" vise à associer un nom logique (représenté par TimeName) à un instant particulier de la journée. De cette façon l'exécution d'un script peut être sous le contrôle de l'horloge temps réel interne du DVP.

Syntaxe : Time TimeName(HH, MM, SS);

HH, MM et SS sont des nombres ou le caractère "*". Ce dernier indique que le champ correspondant ne doit pas être pris en considération dans l'évaluation d'une condition où TimeName intervient.

Exemples :

Time Midi(12, 0, 0);
Time Heure2(*, 30, 0);

Date

L'instruction "Date" vise à associer un nom logique (représenté par DateName) à une date. De cette façon l'exécution d'un script peut être sous le contrôle de l'horloge temps réel interne du DVP.

Syntaxe : Date DateName(JJ, MM, AAAA);

JJ, MM et AAAA sont des nombres ou le caractère "*". Ce dernier indique que le champ correspondant ne doit pas être pris en considération dans l'évaluation d'une condition où DateName intervient.

Exemples :

```
Date Noël(25, 12, *);
Date Date2(20, 2, 2001);
```

Jour de la semaine

Monday,Tuesday, Wednesday…Sunday

L'utilisation d'un "Jour de la semaine" est toujours combinée avec des structures de contrôle de type booléen (WaitEvent, while,if). Ces structures sont détaillées plus loin dans le document.

Exemple :

```
if (Monday)
{
    Clip ClipDuLundi("HEYJUDE.MPG");
    ClipDuLundi.Play();
}
```

Ces entités qui représentent les jours de la semaine offrent un autre niveau de contrôle de l'exécution d'un script en fonction de l'état de l'horloge temps réel interne du DVP.

Input1, Input2, … , Input8

Ces entités représentent l'état logique des 8 signaux d'entrée appelés "déclencheurs d'entrée". Elles sont utilisées pour contrôler le séquencement des actions définies dans les scripts.

Output1, Output2, … , Output4

Ces entités représentent l'état logique des 4 signaux de sortie appelés "déclencheurs de sortie", ces signaux pouvant à leur tour être utilisés pour synchroniser d'autres équipements (DVP ou autres).

Fonctions associées

Une ou plusieurs fonctions sont associées à chacune de ces entités.

Fonctions associées aux entités "Clip" : Play, Loop, Stop, Wait, Prepare, Start

Play (Lecture)

Cette fonction vise à démarrer l'exécution unique d'un clip donné (représenté par ClipName).

Syntaxe : ClipName.Play();

Exemple: alive.Play();

Loop (Boucle)

Cette fonction vise à lire de manière répétée (à l'infini ou un nombre de fois déterminé) un clip donné (représenté par ClipName).

Syntaxe : ClipName.Loop(N);

Lorsque N est spécifié de manière explicite, le clip est lu N+1 fois (N boucles).

Lorsque N n'est pas spécifié, le clip est lu en boucle indéfiniment.

Exemples:

alive.Loop(3); (lit 4 fois le clip "alive.MPG")

lord.Loop(); (lecture continue du clip "lord.VOB")

Stop (Arrêt)

Cette fonction vise à stopper sur le champ l'exécution d'un clip donné (représenté par ClipName). Le DVP procèdera de suite à l'exécution de l'instruction suivante.

Syntaxe : ClipName.Stop();

Exemple:

lord.Stop();

Remarque : la manière dont la lecture a été démarrée (Play ou Loop) est sans importance.

Wait (Attente)

Cette fonction vise à différer l'exécution de l'instruction suivante jusqu'à la fin d'un clip donné (représenté par ClipName) ou jusqu'à la fin de son exécution en boucle (selon la façon dont l'exécution a été lancée).

Syntaxe : ClipName.Wait();

Exemples:

alive.Wait();

lord.Wait();

Prepare et Start

L'instruction "Prepare" vise à préparer un clip pour un démarrage instantané lors de l'exécution d'une instruction de type "Start". Après avoir "préparé" un clip, le décodeur MPEG sera mis en « pause » juste avant l'affichage de la première image. L'utilité de ces instructions est associée aux applications qui nécessitent un démarrage synchrone (précis à la trame près) de plusieurs DVP.

Syntaxe: ClipName.Prepare();
Ou

ClipName.Prepare(paramètre);

ClipName.Start();
Ou
ClipName.Start(paramètre);

Lorsque l'instruction "Prepare" est utilisée sans paramètre (/entité) entre les parenthèses, le script continuera son exécution après la "préparation" du clip. Le clip ne commencera à être joué qu'après l'exécution d'une instruction de type "Start".

Lorsque l'instruction "Prepare" est utilisée avec un paramètre (/entité) entre les parenthèses, le script s'arrête après la "préparation" du clip. Dès que l'entité (/paramètre) devient "True", le clip commence à jouer et le script continuera son exécution. A l'heure actuelle, seulement les déclencheurs d'entrée peuvent être utilisés comme entités valides.

L'instruction "Start" peut aussi être utilisée avec un paramètre entre les parenthèses. Ce paramètre doit être le nom d'une entité de type déclencheur de sortie. Au moment de l'exécution, le déclencheur en question sera activé pendant 250 ms.

Exemples :

Exemple 1 : alive.Prepare();
suivi par
alive.Start();

Exemple 2 : alive.Prepare();
suivi par
alive.Start(Output1);

Exemple 3 : alive.Prepare(Input1);

Les instructions comme celles de l'exemple 2, apparaîssent dans les scripts de type MAITRE, tandis qu'une instruction comme celle de l'exemple 3 est surtout utilisée dans les scripts de type ESCLAVE.

Un DVP "MAITRE" peut ainsi "guider" plusieurs DVP "ESCLAVES".
Il suffit de connecter le "Output1" du MAITRE aux "INPUT1" des esclaves, afin de réaliser un démarrage synchrone.

Fonctions associées aux entités "PlayList " et "PlayListFile" : Play, Loop, Stop, Wait, Prepare, Start

Play (Lecture)

Cette fonction vise à démarrer l'exécution unique d'une liste de reproduction donnée (représentée par ListName).

Syntaxe : ListName.Play();

Exemple:

Hitparade.Play();

Loop (Boucle)

Cette fonction vise à lire de manière répétée (à l'infini ou un nombre de fois déterminé) une liste de reproduction donnée (représentée par ListName).

Syntaxe : ListName.Loop(N);

Lorsque N est spécifié de manière explicite, la liste est jouée N+1 fois.

Lorsque N n'est pas spécifié, la liste est lue indéfiniment en boucle

Exemples:

Hitparade.Loop(3); (lit 4 fois la liste "Hitparade")

Hitparade.Loop(); (lecture continue de la liste "Hitparade")

Stop (Arrêt)

Cette fonction vise à stopper sur le champ l'exécution d'une liste de reproduction donnée (représentée par ListName). Le DVP procèdera de suite à l'exécution de l'instruction suivante .

Syntaxe : ListName.Stop();

Exemple:

Hitparade.Stop();

Remarque : la manière dont la lecture a été démarrée (Play ou Loop) est sans importance.

Wait (Attente)

Cette fonction vise à différer l'exécution de l'instruction suivante jusqu'à la fin d'une liste de reproduction donnée (représentée par ListName) ou jusqu'à la fin de son exécution en boucle (selon la façon dont l'exécution a été lancée).

Syntaxe : ListName.Wait();

Exemple:

Hitparade.Wait();

Remarque : au cas où il s'agit d'une liste représentée par un fichier externe (PlayListeFile), le fichier ".pvp"correspondant sera lu juste avant l'exécution d'une instruction de type "Play" ou "Loop". Ceci permet de modifier le contenu d'un fichier de type "PlayListFile" pendant l'exécution d'un script, sans devoir l'arrêter.

Prepare et Start

Les instructions "Prepare" et "Start" peuvent être utilisées avec les entités "PlayList" et "PlayListFile, de manière similaire à celle appliquée pour les entités de type "Clip".

Fonctions associées aux entités "Timer" : Start, Stop, Wait.

Start (Démarrage)

Cette fonction initialise un temporisateur donné (représenté par TimerName) à l'état défini dans la déclaration du-dit timer et démarre le compte à rebours.

Syntaxe : TimerName.Start();

Exemple:

uneminute.Start();

Remarque : il est possible de réinitialiser un temporisateur à une valeur différente de celle utilisée lors de sa déclaration et de le démarrer, le tout en une instruction unique.

Exemple : uneminute.Start(15);

Cette instruction va réinitialiser le temporisateur à 15 secondes au lieu de 60 secondes et va ensuite démarrer le compte à rebours.

A noter que lors d'une utilisation ultérieure de la syntaxe uneminute.Start(); ce temporisateur sera réinitialisé à 15 secondes.

Stop (Arrêt)

Cette fonction stoppe sur le champ le compte à rebours du temporisateur représenté par TimerName.

Syntaxe : TimerName.Stop();

Exemple: dixsecondes.Stop();

Wait (Attente)

Cette fonction vise à attendre l'expiration du temporisateur représenté par TimerName avant d'exécuter l'instruction suivante du script.

Syntaxe : TimerName.Wait();

Exemple: uneminute.Wait();

Fonction associée aux entités "InputN " : Wait.

Wait (Attente)

Cette fonction vise à attendre l'activation d'un déclencheur d'entrée donné (représenté par InputN) avant d'exécuter l'instruction suivante du script. Pour rappel, le niveau 'bas' est le niveau actif des déclencheurs.

Syntaxe : InputN.Wait();

Exemple: Input4.Wait();

Fonctions associées aux entités de **"**OutputN **"** : Set, Clear

Set (Activer)

Cette fonction vise à activer un déclencheur de sortie donné représenté par OutputN. Pour rappel, le niveau 'bas' est le niveau actif des déclencheurs.

Syntaxe : OutputN.Set();

Exemple: Output1.Set();

Clear (Désactiver)

Cette fonction vise à désactiver un déclencheur de sortie donné représenté par OutputN. Pour rappel, le niveau 'bas' est le niveau actif des déclencheurs.

Syntaxe : OutputN.Clear();

Exemple: Output1.Clear();

Fonction associée aux entités "Time ", "Date " et "Jour de la semaine "

Wait (Attente)

Cette fonction vise à différer l'exécution de l'instruction suivante jusqu'à ce que l'heure, la date ou le jour de la semaine soit atteint.

Exemples :

```
Midi.Wait();
Noël.Wait();
Samedi.Wait();
```

Etat logique associé

Durant l'exécution des scripts, un état logique "True" (Vrai) ou "False" (Faux) est attribué à chacune des entités énumérées ci-dessus. Lors de l'exécution du script, il est possible de tester l'état de ces entités et de conditionner la suite du script à ces états (voir la section "Structures de contrôle" ci-après).

Etat des entités "Clip " :

L'état d'une entité "Clip" est "True" durant la lecture de ce clip ("False" autremement).

Etat des entités "PlayList " :

L'état d'une entité de "PlayList" est "True" durant la lecture des clips de la liste de reproduction ("False" autremement).

Etat des entités "PlayListFile " :

L'état d'une entité de "PlayListFile" est "True" durant la lecture des clips de la liste de reproduction ("False" autremement).

Etat des entités "Timer " :

L'état d'une entité de "Timer" est "True" durant le compte à rebours ("False" autremement).

Etat des entités "**InputN**" :

L'état d'une entité "InputN" est "True" lorsque que le déclencheur d'entrée associé est actif ("False" autremement).

Etat des entités "OutputN " :

L'état d'une entité de "OutputN" est "True" pendant que le déclencheur de sortie associé est actif ("False" autremement).

Etat des entités "Heure ", "Date " et "Jour de la semaine "

L'état de ces entités est "True" si l'heure, la date ou le jour de la semaine correspond à celui de l'horloge temps réel du système du DVP ("False" autrement).

Remarque : soyez particulièrement attentif à l'utilisation du caractère "*".
Voici quelques exemples basés sur l'utilisation de l'heure :

(11, 20, 0)	est True uniquement à 11:20:00 du matin (pas de "*" : les 3 nombres sont donc vérifiés)
(*, 0, 0)	est True durant la premiè seconde de chaque heure (le caractère "*" signifie ici que toutes les heures sont valables)
(*, *, 29)	est True durant une seconde, chaque trentième seconde de chaque minute (seules les secondes sont vérifiées)
(08, *, 29)	est True durant la trentième seconde de chaque minute entre 8 et 9 heures du matin (seules les heures et les secondes sont vérifiées)
(20, 15, *)	est True durant une minute, de 20H15 à 20H16
(13, *, *)	est True entre 13H00 et 14H00 (les minutes et les secondes sont ne sont pas vérifiées)

Structures de contrôle

Les structures de contrôle permettent l'exécution conditionnelle de certaines tâches, une tâche étant définie comme la séquence des instructions incluse entre les {} de l'instruction de la structure conditionnelle. D'une manière générale, dans toutes les structures de contrôle qui suivent, on procède d'abord à l'évaluation d'une condition. Selon l'état de la condition ("True" ou "False"), le système va accomplir (ou non) la tâche afférente.

La condition est une expression logique impliquant n'importe laquelle des entités définies ci-dessus. Les opérateurs permis sont "And", "Or" et "Not". Pour les personnes qui sont habituées à la programmation "C/C++" ces opérateurs peuvent être remplacés par, respectivement, "&&", "||" et "!". Pour la syntaxe, veuillez vous reporter aux exemples ci-après.

Structure de contrôle "WaitEvent"

Syntaxe : WaitEvent(condition);

L'exécution de l'instruction suivante est différée jusqu'à la réalisation (état True) de la "condition".

Exemple:

```
WaitEvent(Input2);

Output2.Set();

uneminute.Start();
```

```
uneminute.Wait();

Output2.Clear();
```

Le script attend que le déclencheur d'entrée n° 2 soit activé. Lorsque la condition est vérifiée, le script va activer le déclencheur de sortie n° 2 pendant une minute.

Structure de contrôle "while"

Syntaxe : while(condition) {tâche}

Les instructions entre {}sont répétées tant que la condition est vérifiée ("True"). Dès que la condition est égale à "False" le script passe à l'instruction suivante.

Exemple:

```
while(Not Input1 And Not Input2)
{
    alive.Play();
    alive.Wait();
}
```

Le clip "Alive" est lu en mode de boucle jusqu'à l'activation de n'importe lequel des déclencheurs d'entrée n°1 ou n°2. Veuillez remarquer que l'état de l'entrée1 et l'entrée2 est pris en compte uniquement à la fin de chaque lecture du clip.

Structure de contrôle "if..."

Syntaxe : if(condition) {tâche}

Si la condition est "True" le script va exécuter une seule fois les instructions de la "tâche". Sinon le script saute directement à l'instruction suivante.

Exemple:

```
If(Input1)
{
    Output1.Set();
}
```

Si Input1 est activé le déclencheur de sortie 1 est activé. Sinon l'état de ce déclencheur reste inchangé.

Structure de contrôle "if...else…"

Syntaxe : if(condition) {tâche1} else {tâche2}

Si la condition est "True" le script va exécuter une seule fois les instructions appartenant à la "tâche1". Sinon ce sont les instructions appartenant à la "tâche2" qui seront exécutées une fois.

Exemple:

```
If(Input1)
{
        alive.Start();
        alive.Wait();
}
else
{
        letstick.Start();
        letstick.Wait();
}
```

Si Input1 est activé le clip "Alive" est joué une fois entièrement. Sinon c'est le clip "letstick" qui est lu une fois entièrement.

Télécharger des fichiers

Des scripts peuvent également être utilisés pour transférer automatiquement des fichiers depuis le disque amovible vers le disque non-amovible interne et vice-versa. Ces opérations sont possibles grâce respectivement aux instructions "System.Put" et "System.Get".

L'emplacement des fichiers sur les lecteurs de disque est implicitement déterminé par leurextension (voir les exemples ci-après). Les extensions de fichier suivantes sont reconnues par le système :

- Fichiers MPEG : ".MPG", ".VOB", ".MP1", ".MP2", ".MPV", "MPA", "M1V" , "M2V" , "M1P" , "M2P" (\SSVP\clips)
- Fichiers de scripts, playlists : ".SVP", "PVP" (\SSVP\scripts)
- Fichiers d'images : ".GIF" (\SSVP\images)
- Fichiers binaires (exécutables) : toutes les autres extensions (\SSVP)

La fonction "System.Put"

Syntaxe : System.Put("FileName.Ext");

Cette fonction va copier le fichier dont le nom est FileName et dont l'extension est Ext du disque amovible vers le disque non-amovible.

Exemple: System.Put("lord.VOB");

Cette instruction va copier le fichier "lord.VOB" (clip) depuis le répertoire du lecteur de disque amovible contenant les clips MPEG vers le répertoire homologue du lecteur de disque dur interne.

La fonction "System.Get"

Syntaxe : System.Get("FileName.Ext");

Cette fonction va copier le fichier dont le nom de fichier est FileName et dont l'extension est Ext du disque non-amovible vers le disque amovible.

Exemple: System.Get("alive.MPG");

Cette instruction va copier le fichier "alive.MPG" (clip) depuis le répertoire du disque dur interne contenant les clips MPEG vers le répertoire homologue du lecteur de disque amovible.

Superpositions graphiques

Il est possible de superposer des graphiques (par exemple des logos) à la vidéo grâce à l'utilisation des fonctions détaillées ci-après. A noter que le format des fichiers contenant le graphique doit obligatoirement être au format "GIF".

La fonction "Image"

Syntaxe : Image ImageName("filename.ext", xpos, ypos, largeur, hauteur);

Cette fonction associe au nom logique ImageName le fichier "filename.ext" ainsi qu'une série de paramètres tels que l'emplacement sur l'écran d'un point de référence de l'image (le coin supérieur gauche de l'image) ainsi que les dimensions de la partie affichée. Ces paramètres numériques sont exprimés en pixels. (xpos,ypos) = (1,1) correspond au coin supérieur gauche de l'écran, xpos ayant trait à la position horizontale de l'image, tandis que ypos en définit la position verticale. Les paramètres largeur et hauteur déterminent la partie de l'image qui sera affichée.

Exemple:

Image Logo("logo.gif", 1, 60, 60, 60);

Cette instruction déclare une image baptisée Logo, dont le graphisme est défini par le contenu du fichier "logo.gif". Seuls les 60 premiers pixels (en partant de la gauche) des 60 lignes supérieures seront affichées à l'écran. Le coin supérieur gauche de cette partie se situera à l'extrême gauche de la ligne 60 (en partant du haut).

Si (largeur,hauteur) = (0,0), le DVP affichera le graphique dans son entièreté.
Attention que si vous spécifiez une hauteur et une largeur inférieures aux dimensions effective, seul le fragment défini sera affiché en superposition. Si la largeur et la hauteur que vous déterminez sont supérieures aux mesures originelles de l'image, l'image sera affichée dans son ensemble, centrée à l'intérieur de la fenêtre spécifiée.

La fonction "Show"

Syntaxe : ImageName.Show();

Cette fonction affiche à l'écran l'image associée au nom logique ImageName. La dimension et le positionnement du graphique dépend des paramètres associés lors de la déclaration de ImageName.

Exemple: Logo.Show();

Cette instruction place l'image déclarée comme "Logo" sur l'écran.

La fonction “Clear”

Syntaxe : ImageName.Clear();

Cette fonction désactive l’affichage de l’image à l’écran.

Exemple: Logo.Clear();

L’image “Logo” disparaît de l’écran.

REMARQUES IMPORTANTES :

Il est possible d’afficher simultanément plusieurs graphiques en incrustation. Toutefois, quoique la totalité de la largeur de l’écran puisse être utilisée, la zone verticale utilisable est limitée ! Cette zone représente environ la moitié d’un écran (294 lignes). La zone où l’utilisateur peut placer ses graphiques est fixée par l’ordonnée ypos de l’image qui est affichée la première à l’écran. La partie utilisable correspond aux lignes allant de ypos à ypos +293.

A noter aussi que le DVP ne dispose que d’une palette de 16 couleurs. Si une image comporte plus de couleurs, le DVP va transformer automatiquement l’image dans la palette de couleur qu’il connaît. Lorsqu’on affiche simultanément plusieurs images à l’écran, la première image affichée définira les 16 couleurs utilisées.

Le noir absolu (RGB = 0+0+0) s’affiche automatiquement comme transparent. Pour empêcher ce phénomène, l’utilisateur doit remplacer ce noir absolu par une nuance de gris sombre , par exemple en remplaçant (0,0,0) par (1,1,1).

Fonctions de commutation Vidéo et Audio

Par défaut les signaux de sortie vidéo sont toujours actifs. Toutefois, certaines des instructions utilisables dans les scripts permettent d'agir sur ceux-ci.

La fonction "Video.Off()"

Syntaxe : Video.Off();

Cette fonction désactive les signaux de sortie vidéo analogiques (CVBS, Y/C, RGB). Cela signifie que le moniteur TV ne reçoit plus de signaux de synchronisation. Cependant le DVP continue son décodage et l'audio demeure disponible. En pratique cette fonction permet la mise en veille forcée de certains types de moniteurs.

La fonction "Video.Black();"

Syntaxe : Video.Black();

Cette fonction force l'affichage d'un écran entièrement noir sur les moniteurs raccordés à l'une des sorties analogiques. Les moniteurs, dans ce cas, ne perdent pas leur synchronisation.

La fonction "Video.On()"

Syntaxe : Video.On();

Cette fonction réactive les signaux de sortie vidéo analogiques.

Exemples de scripts

Remarque : les lignes des scripts commençant par les caractères "//" sont interprétées comme étant des lignes de commentaires. Les neuf premières lignes de commentaires peuvent être affichées en mode OSD : pour ce faire, pointer le clip voulu au niveau du sou-menu "clips" du menu "fichiers", puis choisir l'option "info".

Exemple 1

```
//Example1.svp
//Lecture sans fin d'un clip unique
Clip Alive("alive.MPG");
Alive.Loop();
```

Exemple 2

```
//Example2.svp
// Joue 4 fois les clips d'une Liste de reproduction
Clip Alive("alive.MPG");
Clip Letstick("letstick.MPG");
Playlist Maliste(Alive, Letstick);
Maliste.Loop(3);
```

Exemple 3

```
//Example3.svp
// Lecture sans fin de la première minute de deux clips
        Clip Alive("alive.MPG");
        Clip Letstick("letstick.MPG");
        Timer uneminute(60);
        while(True)
        {
                Alive.Play();
                uneminute.Start();
                uneminute.Wait();
                Alive.Stop();
                Letstick.Play();
                uneminute.Start();
                uneminute.Wait();
                Letstick.Stop();
        }
```

Exemple 4

```
//Example4.svp : démo déclencheurs entrée/sortie

// Démarrage : boucle sans fin HEYJUDE, toutes les SORTIES inactives

// Input1 ➔ lit 30 sec LETSTICK
//          SORTIE1 active, SORTIE3 bascule
// Input2 ➔ lit 30 sec ALIVE
//          SORTIE2 active, SORTIE4 bascule
// Input 3 ➔ stopper et retourner au démarrage
//Lors de la mise en service, le DVP commence à lire en  boucle un "clip
//d'accueil" (clip HEYJUDE).
//Si le déclencheur d'entrée "Input1" s'active pendant l'exécution du clip
//d'accueil, le DVP lance le clip "LETSTICK" .
//Inversement, l'activation de "Input2" démarre la lecture du clip "ALIVE".
//Après 30s du clip sélectionné, le DVP retourne à la boucle d' "accueil".
//Durant la lecture du clip sélectionné, le DVP va ignorer
//l'activation de tous les déclencheurs d'entrée sauf  de "INPUT3".
//Si "INPUT3" est activé, le DVP retourne immédiatement à la
//boucle d'accueil.
//Durant l'exécution de "LETSTICK" le déclencheur de sortie "Output1" est
//désactivé et "Output3" pulse en permanence.
//Inversement "Output2" est désactivé  durant la lecture de
//"ALIVE" et "Output4" pulse en permanence.
//"Output3" et "Output4" sont inactives durant la lecture
//du clip d' "accueil".

Clip Accueil("HEYJUDE.MPG");
Clip Letstick("LETSTICK.MPG");
Clip Alive("ALIVE.MPG");
Timer Trente(30);
while(True)
{
        if(!Accueil)
        {
                Accueil.Play();
        }
        if(Input1)
        {
                Accueil.Stop();
                Letstick.Play();
                Output1.Set();
                Trente.Start();
                While(!Input3 && Trente)
                {
                        if(Output3)
                        {
                                Output3.Clear();
```

```
                }
                else
                {
                        Output3.Set();
                }
        }
        Output4.Clear();
        Letstick.Stop();
        Output1.Clear();
    }
    if(Input2)
    {
        Accueil.Stop();
        Alive.Play();
        Output2.Set();
        Trente.Start();
        while(!Input3 && Trente)
        {
                if(Output3)
                {
                        Output4.Clear();
                }
                else
                {
                        Output4.Set();
                }
        }
        Output3.Clear();
        Alive.Stop();
        Output2.Clear();
    }
}
```

Exemple 5

```
//Exemple5.svp : démo superposition de graphiques

//Boucle sans fin de HEYJUDE

//Affiche en superposition le dessin du DVP

        Image DVP("player.gif" , 26, 385, 0, 0);

        DVP.Show();

        Clip HeyJude("heyjude.mpg");

        HeyJude.Loop();
```

Exemple 6

```
//Exemple6.svp : Exemple de l'utilisation des RTC dans les scripts
//
//Les clips qui sont lus dépendent du jour de la semaine
//Un clip donné (avec un dessin GIF fixé en superposition) est
//affiché chaque jour ouvrable entre 12H00 et 13H00
//La vidéo est désactivée entre 18H00 et 08H00, mais également
//toute la journée de chaque dimanche
//Une fois le script démarré, la première lecture va
//commencer le jour ouvrable suivant à 08H00.

Clip Clip1("song_1.mpg");
Clip Clip2("song_2.mpg");
Clip Clip3("song_3.mpg");
Clip Clip4("song_4.mpg");
Clip Clip5("song_5.mpg");

PlayList Liste1(Clip1,Clip2);
PlayList Liste2(Clip3,Clip4);

Image Logo("player.gif",20,250,0,0);

Time T08H00(8,*,*);
Time Pause(12,*,*);
Time T13H00(13,*,*);
Time T18H00(18,*,*);

Video.Off();
WaitEvent(T08H00 && !Sunday);

while(True)
{
        while(Sunday) {}
        while(Tuesday || Thursday || Saterday)
        {
                Video.On();
                Liste1.Loop();
                WaitEvent(Pause);
                Liste1.Stop();
                Clip5.Loop();
                Logo.Show();
                WaitEvent(T13H00);
                Logo.Clear();
                Clip5.Stop();
                Liste1.Loop();
```

```
                WaitEvent(T18H00);
                Liste1.Stop();
                Video.Off();
                WaitEvent(T08H00);
                Video.On();
        }
        while(Monday || Wednesday || Friday)
        {
                Liste2.Loop();
                WaitEvent(Pause);
                Liste2.Stop();
                Clip5.Loop();
                Logo.Show();
                WaitEvent(T13H00);
                Logo.Clear();
                Clip5.Stop();
                Liste2.Loop();
                WaitEvent(T18H00);
                Liste2.Stop();
                Video.Off();
                WaitEvent(T08H00);
                Video.On();
        }
}
```