Réseau d'éclairage scénique DMX$_{512}$: Digital MultipleX

Objectifs visés :
- ✓ *Le réseau d'éclairage scénique* ***DMX$_{512}$***
- ✓ *Notion d'univers, de « daisy chain », d'effets lumineux*

0. Préambule


Eclairage scénique

Dans les ***bâtiments tertiaires*** *et* ***secondaires***, les techniques modernes de commande d'éclairage sont réalisées via le réseau non propriétaire ***DALI***. Néanmoins, la dynamique de ce réseau reste trop lente par rapport aux exigences des effets de lumière que l'on souhaite faire sur les scènes de spectacle. Ainsi, dans le cadre de la commande de ***l'appareillage scénique*** (*projecteurs RVB, lyres, stroboscopes, machines à fumée, gobo, …)*, les standards utilisés pour piloter ces équipements via des réseaux sont : pour le plus répandu ***DMX$_{512}$*** (*cf. United States Institute for Theatre Technology : DMX512-A - Asynchronous Serial Digital Data Transmission*), pour le plus récent ***Art-Net*** *(cf. http://art-net.org.uk) ou bien encore les standards* ***MIDI***, ***Cobranet****, …*

L'objectif de ce TP sera de mettre en œuvre le standard ***DMX$_{512}$***, premier réseau à s'être imposé dans le monde du spectacle, à l'aide d'une structure de commande basée sur un automate programmable industriel.

1. Le Réseau DMX

DMX est l'acronyme de ***D****igital* ***M****ultiple****X*** qui est la définition d'une ***interface numérique standardisée non propriétaire*** pour les ***équipements scéniques***. Le terme ***512*** fait référence au ***nombre de canaux disponibles*** sur un même réseau (*appelé* ***univers***). La norme prévoit d'utiliser au maximum ***32 systèmes*** de ***16 canaux*** chacun. Un équipement ***DMX$_{512}$*** peut donc utiliser plusieurs canaux (*par exemple, les ballasts RGBW que l'on utilisera nécessitent quatre canaux consécutifs*).

Comme le réseau ***DMX$_{512}$*** est ***unidirectionnel*** *(du maitre vers les équipements uniquement)*, on ne dispose pas sur ce réseau de compte rendu sur les échanges et sur la communication. Par conséquent, ce standard ne doit pas être utilisé pour des spectacles ou des équipements réclamant de la ***sécurité*** (*spectacles pyrotechniques, équipements assurant le gréement de décors, leur levage, …*).

1.1 Caractéristiques Techniques :

- ✓ *Tout équipement propose une entrée* ***DMX IN****(+, -), et une sortie* ***DMX OUT*** *(+, -) (la sortie* ***DMX OUT*** *du dernier équipement sur la ligne doit être connectée à une résistance d'impédance* ***120 Ω*** *)*
- ✓ *Topologie de type* ***Bus*** *avec un câblage série de type «* ***Daisy chain*** *»*
- ✓ *Bus Maitre-Esclaves unidirectionnel*
- ✓ *Transmission série : 8 bits, 2 bits de stop, sans parité (RS 485)*
- ✓ *Vitesse de transmission :* ***250000*** *bit/s*
- ✓ *Longueur des câbles :* ***300*** *mètres entre équipements*
- ✓ *Le* ***pilotage simultané*** *de plusieurs appareils s'effectue en leur attribuant* ***les mêmes canaux***

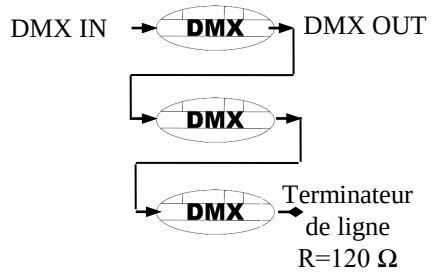
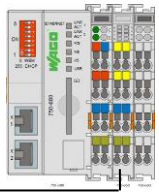


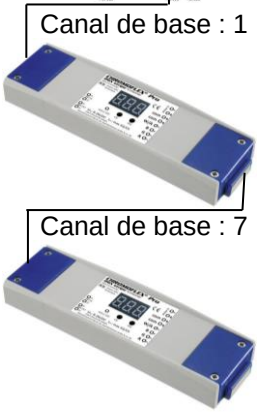
figure 1 : Daisy chain

Une trame ***DMX$_{512}$*** contient dans sa partie utile ***512 octets***, qui par leur position dans la trame définit la valeur de commande des 512 canaux. La norme laisse le soin à chaque constructeur de choisir l'interprétation de ces valeurs (*comprises entre 0 et 255*). Compte tenu de sa vitesse de transmission, le réseau ***DMX$_{512}$*** est donc capable d'envoyer une quarantaine de trames par seconde.

2. Mise en œuvre sur la cible Wago



Pour pouvoir disposer du réseau ***DMX$_{512}$*** sur la structure de commande ***Wago*** au profil ***Ethernet avec ou sans KNX*** (***750-841 ou 750-849***), celle-ci doit être équipée d'un coupleur série ***RS-232/RS485 configurable*** (module ***750-652***). C'est lui qui jouera le rôle de ***maitre DMX***.



D'un point de vue logiciel, on dispose de la bibliothèque « ***DMX_01.lib*** » qui offre plusieurs blocs fonctionnels permettant d'établir la communication et différents effets de lumière (*Color Mixer, Chaser, Fade Sequence, etc …*)

Les équipements ***DMX$_{512}$*** connectés à ce coupleur sont deux ballasts ***RGBW*** (***R**ed, **G**reen, **B**lue, **W**hite*). Ils offrent en face avant deux ***boutons poussoirs*** (***CH-***, ***CH+***) qui permettent de fixer le canal de base de l'équipement. Celui-ci correspond à la sortie ***R***. Les sorties ***G***, ***B***, ***W*** sont associés à ceux qui suivent. Ainsi, si on assigne au ballast comme ***canal de base*** la ***valeur i***, le ***canal i*** désigne sa sortie ***R***, le ***canal i+1*** sa sortie ***G***, le ***canal i+2*** sa sortie ***B***, et le ***canal i+3*** sa sortie ***W***. Quant à l'interprétation des octets de commande associés à ces canaux, ***0*** correspond à l'extinction, ***255*** une pleine puissance.

2.1 Communication

Pour pouvoir envoyer des valeurs sur le bus ***DMX***, il faut faire appel au bloc fonction ***FbDMX_652_Master***. Chaque coupleur ***750-652 DMX master*** doit posséder sa propre instance de ce bloc et elle doit être appelée à chaque cycle afin d'obtenir une communication cyclique. Son interface est donnée par :

```
FUNCTION_BLOCK FbDMX_652_Master
    (* This function block realise the communication to the 750-652 DMX master module *)
    VAR_INPUT
        bCOM_PORT_NR: BYTE:=2;      (* N° of the serial interface used :
                                       1 ⇔ Internal service port,
                                       2 ⇔ 1st connected serial module,
                                       3 ⇔ 2nd connected serial module *)
        iNumberOfChannel: INT:=45;  (* Number of channels to send *)
        xBlackOut : BOOL;           (* TRUE->black out is active *)
    END_VAR
    VAR_IN_OUT
        abDMX_Values : ARRAY[1..DMX_MAX_CH] of BYTE; (* Array of DMX values with
                                                        DMX_MAX_CH=512 *)
    END_VAR
    VAR_OUTPUT
        xReady : BOOL := TRUE;      (* TRUE-> no active transmission,
                                       FALSE->Transmission active, FB busy *)
        bError : BYTE;              (* Error number of  the serial interface cf. Annexe *)
    END_VAR
```

*2.1.1 Proposer une structure de données **CDmxMaster** qui permet d'effectuer une communication cyclique à l'aide d'un coupleur **750-652** configuré en **DMX Master**.*

2.2 Effets de lumière :

*Le bloc fonction **FbRGB_CrossFadeSequence** permet d'effectuer une suite de 10 séquences d'éclairement au maximum sur une source RGB (Red, Green et Blue). Son interface est donnée par :*

```
FUNCTION_BLOCK FbRGB_CrossFadeSequence
    (* Function: This function block is used for setting the color of an RGB light *)

    VAR_INPUT
        xEnable: BOOL;(*Enable the fucntion block. if Black Out then xEnable reset to FALSE*)
        iChannelRed         : INT;          (*RED channel*)
        iChannelGreen       : INT;          (*GREEN channel*)
        iChannelBlue        : INT;          (*BLUE channel*)
        tDelay              : TIME:=t#5s;   (*delay before next colour, Min 1s*)
        xToAndFro           : BOOL;         (*Sequence moves to and fro*)
        iNumberOfColours    : INT:=10;      (*Number of colours to cross fade. [2, 10] *)
        dwColour_1          : DWORD;        (*colour 1 in DWORD format 16#BBGGRR*)
        dwColour_2          : DWORD;        (*colour 2 in DWORD format 16#BBGGRR*)
        dwColour_3          : DWORD;        (*colour 3 in DWORD format 16#BBGGRR*)
        dwColour_4          : DWORD;        (*colour 4 in DWORD format 16#BBGGRR*)
        dwColour_5          : DWORD;        (*colour 5 in DWORD format 16#BBGGRR*)
        dwColour_6          : DWORD;        (*colour 6 in DWORD format 16#BBGGRR*)
        dwColour_7          : DWORD;        (*colour 7 in DWORD format 16#BBGGRR*)
        dwColour_8          : DWORD;        (*colour 8 in DWORD format 16#BBGGRR*)
        dwColour_9          : DWORD;        (*colour 9 in DWORD format 16#BBGGRR*)
        dwColour_10         : DWORD;        (*colour 10 in DWORD format 16#BBGGRR*)
    END_VAR

    VAR_IN_OUT
        abDMX_Values        : ARRAY [1..DMX_MAX_CH] OF BYTE;    (*DMX channel values*)
    END_VAR

    VAR_OUTPUT
    END_VAR
```

*2.2.1 Proposer une structure de données **CRgbCrossFadeSeq** qui permet d'effectuer une séquence de couleur RGB sur un ballast **DMX** à l'aide d'une instance du bloc fonctionnel **FbRGB_CrossFadeSequence**.*

*2.2.2 A l'aide de ces structures de données, établir la programmation en **langage ST** d'un module **DmxPrg** qui a pour but de faire une séquence de **10 couleurs** sur un ruban de leds (**R**, **G**, **B**). Ce ruban est piloté à l'aide d'un ballast électronique **DMX** dont on configurera le canal de base à **7**. La durée entre chaque couleur devra être de 3 secondes et les spécifications de la séquence sont données par le tableau ci-après :*

B	*16#00*	*16#00*	*16#00*	*16#00*	*16#00*	*16#08*	*16#10*	*16#20*	*16#40*	*16#80*
G	*16#08*	*16#10*	*16#20*	*16#40*	*16#80*	*16#80*	*16#40*	*16#20*	*16#10*	*16#08*
R	*16#80*	*16#40*	*16#20*	*16#10*	*16#08*	*16#00*	*16#00*	*16#00*	*16#00*	*16#00*

***Nota** : afin d'optimiser la communication sur le réseau **DMX** avec la structure automate dont nous disposons, il convient de limiter le nombre de canaux à émettre à **15**.*

3. Programmation

*3.1 Récupérer sous U:\Documents\DUT\GEII\ModuleS3\R3.13_EME\Squelettes\Dmx le fichier **750_849.pro**. Ce dernier assure la configuration matérielle et logicielle de la cible **API** disponible pour ce contrôle, ainsi que l'animation de la colonne lumineuse en fonction des boutons poussoirs.*

*3.2 Lancer le logiciel **CoDeSys V2.3** à l'aide du raccourci disponible sur le **bureau** ou **via le menu** Programme -> Wago Software-> CoDeSys for Automation Alliance -> CoDeSys V2.3-> CoDeSys V2.3)*

*3.3 Sous **CodeSys**, ouvrir le squelette de l'application pour ce TP (option **Ouvrir** du menu **Fichier**).*

3.4 Onglet ***Type de données** :*

- ✓ *Déclarer les types **CDmxMaster** et **CRGBCrossFadeSequence***
- ✓ *Les renseigner conformément à votre analyse.*

3.5 Onglet ***Modules** :*

- ✓ *Compléter l'interface du module **DmxPrg** permettant d'établir l'animation souhaitée sur une source Red, Green, et Blue.*
- ✓ *Renseigner son énoncé à l'aide des objets déclarés précédemment.*

*3.6 Compiler votre application, la transférer dans le contrôleur puis vérifier son fonctionnement. Vous disposez dans la salle des quatre stations (**WAGO_DMX_i** avec i=5, 6, 7, 8).*

*3.7 Modifier le programme précédant afin d'obtenir un nouvel effet à l'aide du bloc fonctionnel **FbRGB_RecallColourPalette**. On construira d'abord la classe **CRGB_RecallColourPalette** avant d'en créer une instance pour réaliser la nouvelle animation sur les canaux 1, 2 et 3. L'animation devra permettre de représenter sous la forme d'un mélange de couleurs la valeur de la tension aux bornes du potentiomètre disponible au niveau du pupitre disponible à l'aide du paramètre **bValue** (valeur comprise entre [0,255])*

```
(* Recall up to 10 colours and assigned its value to addressed RGB) *)
FUNCTION_BLOCK FbRGB_RecallColourPalette
    VAR_INPUT
        adwColourPalette    : ARRAY [1..10] OF DWORD;    (*Colour mixture in DWORD
                                                          format 16#BBGGRR*)
        iChannelRed         : INT;          (*RED channel*)
        iChannelGreen       : INT;          (*GREEN channel*)
        iChannelBlue        : INT;          (*BLUE channel*)
        xRecallColour_1     : BOOL;         (*recall colour 1 *)
        xRecallColour_2     : BOOL;         (*recall colour 2 *)
        xRecallColour_3     : BOOL;         (*recall colour 3 *)
        xRecallColour_4     : BOOL;         (*recall colour 4 *)
        xRecallColour_5     : BOOL;         (*recall colour 5 *)
        xRecallColour_6     : BOOL;         (*recall colour 6 *)
        xRecallColour_7     : BOOL;         (*recall colour 7 *)
```

```
        xRecallColour_8     : BOOL;          (*recall colour 8 *)
        xRecallColour_9     : BOOL;          (*recall colour 9 *)
        xRecallColour_10    : BOOL;          (*recall colour 10*)
END_VAR

VAR_IN_OUT
        abDMX_Values        : ARRAY [1..DMX_MAX_CH] OF BYTE;    (*DMX channel values*)
END_VAR

VAR_OUTPUT
END_VAR
```

Annexe *:*

bError	*Error number of the serial interface*
16#00	*No error*
16#01	*This library is not supported by the firmware.*
16#02	*COM port outside of valid range*
16#03	*This function block entity is not yet assigned to a COM port*
16#04	*This function block entity is assigned to a different COM port*
16#05	*COM port already open*
16#06	*COM port already closed*
16#07	*COM port not opened*
16#08	*A write operation is still active (COM1)*
16#09	*These transfer parameters are not supported by the COM port*
16#0A	*Current I/O module settings could not be read*
16#0B	*This library version does not support temporary setting of communication parameters*
16#0C	*I/O module could not be initiatlized*
16#0D	*Error during writing of data to the I/O module FIFO memory*
16#0E	*Contents of FIFO memory were not sent (continuous sending)*
16#0F	*Internal error*

2.1.1 *Proposer une structure de données **CDmxMaster** qui permet d'effectuer une communication cyclique à l'aide d'un coupleur **750-652** configuré en **DMX Master**.*

```
TYPE CDmxMaster:
     STRUCT
          (* données membres *)
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          (* fonction membre *)
          ……
          ……
          ……
     END_STRUCT
END_TYPE
```

*2.2.1 Proposer une structure de données **CRgbCrossFadeSeq** qui permet d'effectuer un séquence de couleur RGB sur un ballast **DMX** à l'aide d'une instance du bloc fonctionnel **FbRGB_CrossFadeSequence**.*

```
TYPE CRgbCrossFadeSeq:
     STRUCT
          (* données membres *)
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          (* fonction membre *)
          ……
     END_STRUCT
END_TYPE
```

2.2.2 *A l'aide de ces structures de données, établir la programmation en **langage ST** d'un module **DmxPrg** qui a pour but de faire une séquence de **10 couleurs** sur un ruban de leds (**R**, **G**, **B**). Ce ruban est piloté à l'aide d'un ballast électronique **DMX** dont on configurera le canal de base à **7**.*

```
PROGRAM DmxPrg
VAR_IN
    bValue:BYTE;
END_VAR
VAR
    dmxMaster:CDmxMaster:=(    ……
                               ……
                               ……
                               ……
                               ……);
    rgbCrossFadeSeq:CRgbCrossFadeSeq:=……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……
                                      ……);
    ……
    ……
END_VAR
(* début *)
……
……
……
……
……
……
……
……
……
```

```
……
……
……
……
……
……
……
……
……
……
……
……
……
```

*2.2.3 Modifier le programme précédant afin d'obtenir un nouvel effet à l'aide du bloc fonctionnel **FbRGB_RecallColourPalette**. On construira d'abord la classe **CRGB_RecallColourPalette** avant d'en créer une instance pour réaliser la nouvelle animation sur les canaux 1, 2 et 3. L'animation devra permettre de représenter sous la forme d'un mélange de couleurs la valeur de la tension aux bornes du potentiomètre disponible au niveau du pupitre disponible à l'aide du paramètre **bValue** (valeur comprise entre [0,255]).*

```
TYPE CRgbRecallColourPalette:
     STRUCT
          (* données membres *)
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          ……
          (* fonction membre *)
          ……
          ……
     END_STRUCT
END_TYPE
```

```
PROGRAM DmxPrg
VAR_IN
    bValue: BYTE;
END_VAR
VAR
    dmxMaster:CDmxMaster:=( (* idem que précédemment *) );
    rgbColourMixer789:CRgbColourMixer:=( …… );
    (* Réservation pour l'animation Fade Sequence*)
```

```
    (* Animation Fade Sequence *)
```