MiCom_v2022

1.35

Table des matières

MiCom_v202x est une bibliothèque de fonctions facilitant la mise en œuvre des protocoles de communication supportés par les modules Midi Ingénierie (HMS) suivants : MIBL, MIP, BMAC, DMAC, µMAC, MAC23, MAC34, SM4F

Cette bibliothèque se présente sous la forme d'un assembly .Net FrameWork directement intégrable dans les développements Windows C# ou LabVIEW (voir exemples en annexe)

L'assembly est distribué en architecture x64, et AnyCpu . la forme x86 reste disponible sur demande (mail).

Cette documentation décrit les appels aux méthodes et propriétés de l'assembly nécessaires pour mettre en place un dialogue sur un module Midi Ingénierie.

Le lecteur devra néanmoins appréhender la documentation du module Midi Ingénierie en sa possession pour connaitre la syntaxe et les contraintes des différentes commandes.

Un utilitaire (AppMiCom_v2025.exe) est également disponible comme interface avec Python, Excel, …
Il est accessible dans le dossier App_MiCom du package d'installation.

l'assembly est disponible depuis le site internet Midi-Ingenierie.com . Il se présente sous la forme d'un package .zip qui peut être dézippé sur n'importe quel dossier.

Les liens utilisés pour cette documentation (section Exemples) sont basés sur le dossier cible
c:\Hensoldt Mechatronic Solutions\MiCom_v2022

1) Méthodes

Method	Signatures / Usage	informations
Open	Ouverture de la ligne de communication int **Open**(string *drivermidi* , string *portname* , string *baudrate*) autres signatures possibles : int **Open**(string *drivermidi* ,string *portname*) int **Open**(string *drivermidi*) int **Open**() La valeur en retour est le **Status** MiCom_v2022	Cette méthode est nécessaire avant tout envoi de commande Les formes réduites **Open**() utilisent les properties concernées en lieu et place des paramètres d'appel de la forme complète : Un appel à la forme réduite **Open()** sera transformé en appel à Open(DriverMidi,PortName,BaudRate) par MiCom_v2022.
Send	Emission d'une commande int **Send**(string *commande*) La *commande* est transformée en trame série en fonction du protocole imposé par le *drivermidi*. L'émission de la trame se fait sur le Com *portname* en respectant le baud rate *baudrate.* L'analyse de la trame en retour du driver positionne les properties *Status* et *Reponse* La valeur retournée est le ***Status*** MiCom_v2022	*La property **Reponse** fournit la réponse du Driver si la commande est une requête* ***DetailCr** détaille l'erreur quand Status.**Recept_Nack or failed** est positionné* ***X_Etat** est significatif si le driver est compatible XOFF_étendu*
Close	Libération du port COM utilisé pour la communication Int Close() La valeur retournée est le *Status* MiCom_v2022	Les échanges (Send ou Dialog) sont encadrés par **Open** / **Close** Pour changer un paramètre de communication, il faut effectuer un **Close** puis un **Open** avec les nouvelles valeurs
Dialog	Envoi d'une commande et récupère la **Reponse** int **Dialog**(string *commande* , string& *reponse*)	Remplace les écritures : Cr = miCom.**Send**(requete) Retour = micom.**Reponse**
Get_Info	Retour d'informations string **Get_Info**(string *about*) *about* : définit le type d'information retournée par Get_info	about : 'IDENTITY' : version du MiCom_v2022 'DRIVER' : liste des drivers MIDI possibles 'PORT' : liste des COMs installés sur la machine 'BAUD' : liste des baudrates par DRIVER 'ERROR_MACCONVERTER' : détail de l'erreur MAC_CONVERTER

2) Propriétés

Property	type	usage
Identity	*string*	*information version du MiCom_v2022*
Status	integer	compte rendu des méthodes / propriétés
PortName	string	port COM utilisé comme support de communication
DriverMidi	string	le driver détermine le protocole et la syntaxe des échanges
BaudRate	string	fixe le baud rate, ce paramètre doit correspondre à celui du driver.
IsOpen	boolean	précise si un port COM est 'ouvert', (utilisé), par MiCom_v2022
Reponse	string	Information retournée par le driver suite à une requête
DetailCr	integer	complément du Status lorsque le flag **Recept_Nack or failed** est positionné
X_Etat	integer	Retour du X_Etat pour les drivers compatibles protocole XOFF ETENDU
TimeOut	integer	délai d'attente d'un retour driver, avant de déclarer une absence de réponse.

3) Open()

int status = **open**(string driver, string port , string baud)

La méthode Open, initialise les structures internes nécessaires à la mise en œuvre de la communication, avec les paramètres suivants :

- ✓ **driver**, l'identifiant d'un DRIVER Midi Ingénierie présent dans la liste suivante :
  (**MIBL , MAC23,MAC34, DMAC, BMAC MICROMAC17, MIP**)

- ✓ **port ,** une référence à un identifiant de PORT disponible* sur la machine

- ✓ **baud** , la valeur du baud rate de la communication

  Attention : le baud rate précisé ici fixe UNIQUEMENT un baud rate coté émetteur (le Pc)
  (réf Manuel utilisateur du driver pour la syntaxe de la commande de modification du baud rate sur le driver)

En retour, la méthode **Open** positionne le **Status** avec les valeurs suivantes possibles :

0x8010 L'information DriverMIDI n'est pas un élément de la liste

0x8008 L'information PortName est invalide

0x8020 Le Baudrate n'est pas une valeur numérique acceptable

- *Les erreurs multiples sont possibles (0x8018 , 0x8028 ,0x8030 et 0x8038)*

0x8100 Impossible de finaliser la fonction OPEN :

- Le PortName est déjà attribué à une autre application
- Le PortName n'existe pas en tant que PortCOM (ex BLUETOOTH ?)

La méthode Open met à jour les propriétés suivantes : DriverMidi , PortName , BaudRate

La méthode Open se décline également avec les signatures suivantes :
int status = open(string DriverMidi, string PortName)
int status = open(string DriverMidi)
int status = open( )

les paramètres doivent être renseignés au préalable par les properties .
exemple :

```
//  myCom est l'objet créé par le new MiCom_v2022()   c#
myCom.PortName = "COM9" ;
myCom.DriverMidi = "MIBL"  ;            // sélection d'un driver
```

☞ *la sélection d'un driver imposera **toujours** pour la property BaudRate la valeur à défaut du baud rate pour ce driver : implicitement myCom.Baudrate est initialisée avec 115200 (baud rate à défaut pour le driver MIBL)*

```
myCom.Open() ;  ➔ équivaut donc à Open("MIBL", "COM9","115200")
//
//ici l'émission / échange par la méthode Send ou Dialog
//
myCom.Close() ; ➔ fermeture de la ligne COM9

myCom.PortName = "COM10" ; // ➔modification du PortCom pour un autre driver MIBL

mycom.Open() ;
```

4) Send()

int Send(string Cmde)

La méthode Send gère de bout en bout l'émission de la commande Cmde ainsi que la réception du retour d'information du drivermidi .

- ✓ **Cmde** , une commande destinée à un driver Midi-Ingénierie

Tous les éléments nécessaires à l'échange sont renseignés au préalable par la méthode **Open** associée aux properties DriverMidi,Portname , Baudrate

Cette méthode positionne les properties suivantes :

Status, qui est également la valeur en retour de la méthode Send

Reponse, qui mémorise le retour d'information lorsque la commande émise est une requête

la fonction **Send** positionne le **status** avec les valeurs suivantes possibles :

0x8100	La méthode Open n'a pas été faite (incorrecte)
0x0000	la commande à été correctement interprétée par le DriverMidi
0x0001	l'information en retour du driver est disponible dans Reponse

Voir Détail Status MiCom_v2022 pour les autres valeurs possibles.

La commande sera convertie en message en fonction du protocole supporté par le DriverMidi

5) Get_Info()

string Get_Info(string about)

La méthode Get_Info() permet de rapatrier les informations suivantes :

About :	
identity	L'identification de l'assembly MiCom_v2022 . identique à la property Identity
driver	Liste des drivers supportés par MiCom_v2022 pour la version 1.35 : MIBL DMAC BMAC MICROMAC17 MAC23 MAC34 MIMU MIP Les identifiants drivers sont séparés par le caractère ESPACE (0x20)
port	Liste des ports tels que définis dans le registre HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM
baud	Liste pour chaque driver, les valeurs possibles de baud rate extrait : MIBL=115200;BMAC=38400 9600 19200 115200;… Le caractère ';' sépare les groupes drivers Les valeurs de baud rate sont séparés par le caractère ' ' (ESPACE = 0x20) Le caractère '=' sépare un identifiant de driver de sa liste de baud rate Le baud rate à défaut est toujours présenté en début de liste. Le driver s'initialise sur cette valeur à défaut en sortie usine ou à chaque commande dite de 'retour en configuration usine ' C'est également sur cette valeur de baud rate que s'initialise la proterty BaudRate à chaque modification de la property DriverMidi

6) Status ou Cr

Le Status est commun à toutes les méthodes et propriétés.
C'est également la valeur de retour pour les méthodes Open , Send ,Close ,Dialog . Il est alors confondu avec le Cr

de type integer sur 16bits utiles, il est organisé comme suit :

b15	Fail / Error
b14	com lost
b13	**recept timeout**
b12	emis failed
b11	**recept nack or failed**
b10	control
b09	out of range
b08	open send
b07	close
b06	mac converter error
b05	baudrate
b04	driver
b03	port
b02	timeout
b01	mac converter
b00	recept from driver

Information :

recept from driver : la commande générée est une requête, le retour d'information est correctement reçu
mac converter : la commande générée est destinée à un module Midi de type MAC23 ou MAC34

Défaut :

mac converter error : la conversion de la commande en mode EXPERT MAC23/MAC34 a échoué
baudrate : valeur négative ou non numérique
driver : la référence fournie est inconnue
port : la référence est nulle ou non conforme
timeout : valeur out of range

Warning :

recept timeout : absence d'acquittement (Câble débranché, défaut d'adresse, erreur de baud rate,…)
recept nack or failed : commande refusée par le driver (**NACK**) ou la trame retour incorrecte (**FAILED**)

com lost : la communication n'est plus opérationnelle, le nb de timeOut dépasse un seuil fixé ou **emis failed**.
emis failed : émission de la commande en erreur (le port COM ?)
control : erreur trame est de type CHECKSUM
out of range: erreur est de type NB CARACT
open send : générée par la fonction Open (port déjà utilisé ou absent) ou par la fonction Send (absence du Open)
close : générée lors de la fonction Close

Fail / Error:

Ce flag est automatiquement positionné dés qu'un flag autre que Recept from driver / MacConverter est positionné. Il n'est jamais positionné SEUL, la valeur 0x8000 n'existe pas :
un Status > 0x8000 signale une erreur

7) Exploitation du Status

b15	b14	b13	b12	b11	b10	b09	b08	b07	b06	b05	b04	b03	b02	b01	b00
Fail /Error	com lost	recept timeout	emis failed	recept nack or failed	control	out of range	open send	close	mac converter error	baudrate	driver	port	timeout	mac converter	recept from driver

Les Cr suivants sont inférieurs à 0x8000 : la communication est correcte

0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Nominal hors requête (driver autre que MAC34/MAC23)
0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Nominal pour requête (driver autre que MAC34/MAC23)
0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	MAC34 / MAC23 : nominal hors requête
0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	MAC34 / MAC23 : nominal pour requête

Les Cr supérieurs à 0x8000 signalent un défaut de communication ou une configuration incorrecte :

0x8004	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	valeur de timeout hors limite
0x8008	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	portname non valide [4]
0x8010	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	driver inconnu
0x8020	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	baudrate incorrect
0x8100	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	erreur : Aléa[1] sur fonctions Open ou Send
0x8800	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	défaut commande[2], erreur de syntaxe
0xA000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	absence de réception (TimeOut)
0xE000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	perte de la communication [3] (trop de timeout consécutifs)

1 : La fonction **Open** positionne cette erreur dans le cas ou le **port** précisé en appel est déjà en cours d'utilisation par une autre application, si ce port n'existe pas (COM55 pour COM5) ou si ce port valide est un port pour l'usage Bluetooth .
La fonction **Send** positionne cette erreur dans le cas ou il n'y a pas eu d'appel valide à la fonction **Open** AVANT l'appel à la fonction **Send**

2 : Cette erreur signale une réponse non nominale de la part du driver. (Voir Annexe)
En l'absence des erreurs **control** ou **out of range**, le détail de l'erreur est exploitable par la fonction Get_Info(« DETAIL_CR »)

3 : Cette erreur signale une « perte » de la communication :
0xE000 : combinée avec **recept timeout** → le nombre de timeout consécutif dépasse un seuil fixé (15)
0xD000 : combinée avec **emis failed** → les émissions sont impossibles (?? *aléa grave sur l'uart*)

4 : Port invalide : L'information **port** peut être fixée de 2 façons :

a) Indépendamment de la fonction Open, la fonction Open se faisant alors par une signature réduite :

```
Midi_Ingenierie.MiComm_2022 myCom = new Midi_Ingenierie.MiComm_2022();
myCom.PortName = "COM10";   // sélection du Port ,indépendament de la fonction OPEN
//....
myCom.Open("MIBL")      ;   // fonction OPEN , signature réduite
```

b) Directement par la fonction Open (signature complète):

```
Midi_Ingenierie.MiComm_2022 myCom = new Midi_Ingenierie.MiComm_2022();
myCom.Open("MIBi","COM10","115200")      ;   // fonction OPEN signature complète
```

Dans tous les cas, l'erreur ***Port*** est positionnée si l'information PortName est nulle ou non conforme.
L'erreur ***Open Send***, est positionnée lors de la fonction **open,** la présence du COM est vérifiée et validée :
➔ un COMxx de type VCP doit être installé AVANT l'appel à la fonction **Open** (*driver* , COMxx,..)
Open Send sur la fonction **send** signale l'absence du port (désinstallé entre le **open** et le **send**)

8) Detail_Cr

DetailCr est pertinent lorsque l'erreur **Status.Recept Nack or Failed** est positionnée
La connaissance des protocoles de communication est nécessaire pour la compréhension du DetailCr

Trame incohérente	b15		
	b14	Bad_XON	La trame reçue n'est pas conforme au protocole retenu : • Les caractères de protocole (ACK , XOFF, XON , ..) sont mal placés ou absents . • Présence de caractères NULL (0x00). • Le nombre de caractères reçus est hors limite **Il est probable que l'échange en cours soit déclaré en TimeOut (Status.b13)*
	b13	Bad_XOFF	
	b12	XON_as_XOFF	
	b11	Bad_ACK	
	b10	XON_as_ACK	
	b09	Overflow*	
	b08	Null	
Trame cohérente	b07		
	b06		La trame reçue est conforme au protocole retenu mais son contenu signale les erreurs suivantes :
	b05	**Error Cmde**	ErrorCmde (b05), ou XOFFERREUR, erreur d'interprétation (DMAC, BMAC, µMAC)
	b04	**Error Cmde**	ErrorCmde (b04), ou XONERREUR, erreur d'interprétation (MAC23, MAC34)
	b03		
	b02		
	b01		
	b00	**NACK**	NACK (b00), la commande reçue par le module n'est pas conforme au protocole

Les trames incohérentes systématiquement sont surtout la conséquence d'une liaison série mal câblée (niveau de bruit trop important)
La trame peut être incohérente lorsque 2 modules sont configurés sur la même adresse et répondent en même temps à une requête (position, version, …)

L'erreur Error_Cmde est générée par un module MIDI dans les cas suivants :

Commande inconnue
Paramètre hors limite
Commande refusée par le module en fonction de son état , ou activité

9) X_Etat

Le caractère X_Etat est un élément du protocole de communication pour les drivers de la famille DMAC

Il est renvoyé par le module (DMAC, BMAC, µMAC) pour signaler :

- Une erreur d'interprétation, il est alors considéré comme un XOFF_ERREUR au niveau protocole. Pour MiComv2022, le bit11 du **Status**, *recept nack or failed*, ainsi que le bit05 du **DetailCr** sont positionnés

- En l'absence d'erreur, le X_Etat signale les informations suivantes :

b07	**1**	
b06	**warning**	se reporter au manuel utilisateur de chaque module
b05	**disjonction**	température, sous tension, surtension,…
b04	réservé	
b03		
b02	**busy**	moteur non asservi ou position non atteinte
b01	**move**	moteur en mouvement
b00	**power**	moteur sous tension (power on /power off)

Un exemple d'utilisation de la propriété X_Etat est disponible dans la section Example du MiCOm_v2022

Précaution :
Le X_Etat renvoyé par le module reflète son état en début d'interprétation/exécution de la commande reçue.

En supposant le module à l'arrêt, une commande de mouvement telle que 00MOVE_ON 10000 va rapatrier un X_Etat signalant … que le module est à l'arrêt.
Une autre commande, telle une requête de relecture 00READ #POSITION, en plus de rapatrier la position, signalera l'état en mouvement

Annexes

constantes pour la description du Status

les constantes ci-dessous, définies en C# , sont utilisées dans les exemples fournis

```
const int b__RepInTrame__    = 0x0001 ; // b0  il existe une réponse dans la trame retour du driver
const int b__MacConverter__  = 0x0002 ; // b1  commande convertie en mode EXPERT par le MacConverter
const int b__DefTimeOut__    = 0x0004 ; // b2  le paramètre timeout est hors limite
const int b__DefPort__       = 0x0008 ; // b3  le paramètre 'port name' n'est pas valide
const int b__DefDriver__     = 0x0010 ; // b4  le paramètre driver n'est pas une référence de Driver connue
const int b__DefBaudrate__   = 0x0020 ; //  b5  le paramètre baud est incorrect
const int b__ErrMacConverter= 0x0040 ; // b6  erreur MacConverter : commande inconnue
const int b__Close__         = 0x0080 ; // b7  erreur sur la fonction Close, assimilé à un défaut SYSTEM
const int b__OpenSend__      = 0x0100 ; // b8  erreur générée par la fonction Open ou Send (port?)
const int b__OutOfRange__    = 0x0200 ; // b9  défaut de trame, champs LNG  (perte de caractère)
const int b__Control__       = 0x0400 ; // b10 défaut de trame, champs CHK  (checksum)
const int b__RecepFailed__   = 0x0800 ; //  b11 la trame retour est un NON ACQUITTEMENT ou WARNING
const int b__EmisFailed__    = 0x1000 ; // b12 Emission impossible = défaut SYSTEM
const int b__RecepTimeOut__ = 0x2000 ; // b13 le TimeOut s'est déclenché : pas de réception
const int b__ComLost__       = 0x4000 ; // b14 Timeout permanent / Perte de ligne = défaut SYSTEM
const int b__Fail_ERROR__    = 0x8000 ; // b15 Status Error
```

un modèle d'analyse du Status pourrait être :

```
myCom.Send("00MOVE_TO +78747 ") ;   //une commande
Cr = myCom.Status   ;                       // prise en compte du Status

if (Cr > b_Fail_ERROR)                  // signale un aléa, une erreur
{
    If   ( (Cr   &  b__RecepFailed__) ==  b__RecepFailed__)
    {
        // la commande n'est pas acceptée ou interprétée par le module
    }
}
```

Exemple d'utilisation : C#

Plusieurs exemples de mise en œuvre du MiCom_2022 sont disponibles dans le package d'installation.
Les exemples sont dédiés à un driver particulier (ex MIBL, MICROMAC, MAC23...).
Pour chacun d'entre eux, une forme exécutable est disponible, le lien donné dans le tableau ci-dessous suppose que le package d'installation soit dézippé dans le dossier :
C:\Hensoldt Mechatronic Solutions\MiCom_v2022

Exemple	driver	Détail
HMS_MiCom2022_MIBL	MIBL	Lecture du registre R00
HMS_MiCom2022_MicroMAC	µMAC	MOVE_ON , POWER OFF, REA #POSITION , extraction position Check du Status , et du Detail_CR Génération d'un mouvement
HMS_MiCom2022_MicroMAC_XETAT	µMAC	Identique à HMS_MiCom2022_MicroMAC mais la gestion des fins de mouvement se fait par l'exploitation du X_ETAT
HMS_MiCom2022_MAC23_MAC34	MAC23 MAC34	Commande standard Reset : mr Couple : gi 20 Mvt vitesse : gf xxxxx Arrêt : ge Power off : gr
HMS_MiCom2022_MACEXPERT	MAC23 MAC34	Commande mode expert Reset : 000000000000 Couple : 0C000000001A Mvt vitesse : 150000000xxx Arrêt : 170000000000 Power off : 1A0000000000
HMS_MiCom2022_DMAC	DMAC	Reset : 00MRE Couple : 00#TORQUE_RATIO xx Mvt vitesse : 00MOVE_SPEED xx Arrêt : 00STOP Power off : 00POWER OFF

HMS_MiCom2022_MIBL

méthodes Open, Send,
propriétés Status, Reponse
lecture du registre R01,
émission d'une mauvaise commande,

l'appelant doit fournir un ComName valide ou le préciser au RunTime

La ligne d'appel précise le ComName a utiliser pour le dialogue PC ←→ MIBL

```
c:\MiCom_v2022\Package\examples\MIBL>HMS_MiCom2022_MIBL.exe COM5
>HMS_MiCom2022_MIBL.exe  ==> how to use MiCom_v2022 for MIBL driver
>Usage : HMS_MiCom2022_MIBL.exe   "ComName"

>here after , myCom is the instance type of MiCom_2022.

>first check  : version and copyright
myCom.Get_Info('IDENTITY') = MiCom_2022 v1.35 ©Midi-Ingenierie 07/2025

>looking for the comName arg in the command line
>COM5 found , so ComName = COM5

>method Open = select DRIVER,PortCOM and baudrate
myCom.Open('MIBL','COM5',115200') = 0x0000

>method Send  = send a command / requete to the driver (MIBL)
myCom.Send('R00')  Cr= 0x0001
>property Reponse = the last driver (MIBL) response
myCom.Reponse = 0000F50003

myCom.Send("X00") a bad command !
>property Status = the last Status
myCom.Status  = 0x8800
myCom.Reponse =

[ESC] All is done ,close the program
```

Le Status est > 0x8000 : **ERREUR**

Le module MIBL a signalé l'erreur dans la trame retour et MiCom_v2022 le remonte à l'appelant dans le Status (bit11)

HMS_MiCom2022_MicroMAC

- …\MiCom_v2022\Package\examples\MICROMAC17>*HMS_MiCom2022_MicroMac.exe*

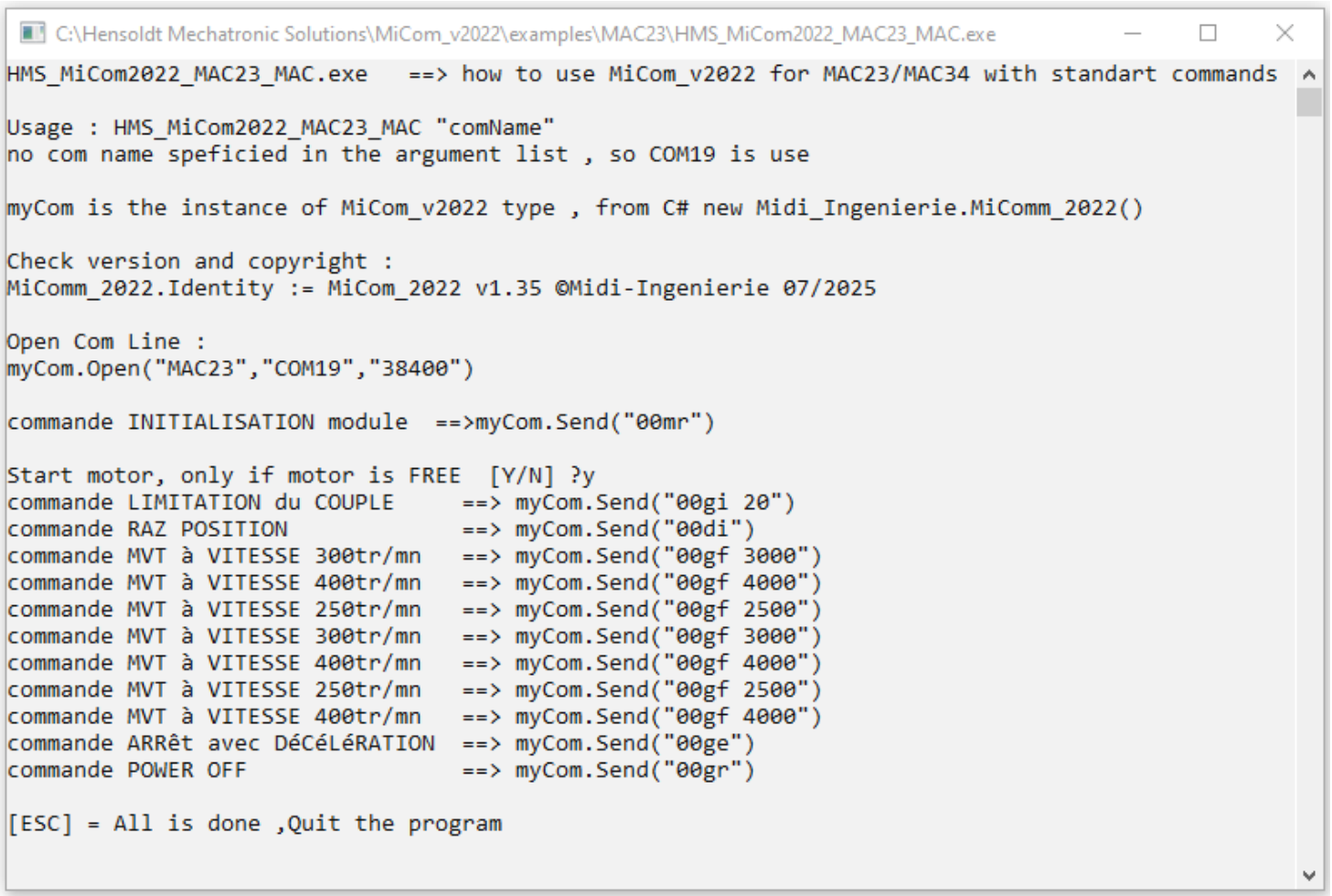
```
C:\Hensoldt Mechatronic Solutions\MiCom_v2022\examples\MICROMAC17\HMS_MiCom2022_MicroMac.exe
HMS_MiCom2022_MicroMac.exe   ==> how to use MiCom2022 for MICROMAC17 brushless digital axis
Usage : HMS_MiCom2022_MicroMac.exe  "ComName"
myCom is the instance of MiCom_v2022 type , from C# new Midi_Ingenierie.MiComm_2022()

Check version and copyright :
MiComm_2022.Identity := MiCom_2022 v1.35 ©Midi-Ingenierie 07/2025

no ComName ref in command line argument : input a ComName >com19
Open  , select DRIVER , PortCOM , and baudrate   all as string (C#)
myCom.Open("MICROMAC17", "COMxx", "38400")  => 0000

Send  , send a command / requete to the MICROMAC17 driver , adress 00
myCom.Send("00RV")  => 0001
Cr=0x0001 ==> a reponse from the MICROMAC17 exist : myCom.Reponse
myCom.Reponse = 00EV v5.0 9170 "MI_uMAC17_9170-00492_" 0.5

Start motor, only if motor is FREE  [Y/N] ?y
The start command :
myCom.Send("00MOVE_ON +10000")  Status= 0000

Get POSITIONs:  (5 cmdes 00REA #POSITION )
myCom.Send("00REA #POSITION")  :00#POS= 4
myCom.Send("00REA #POSITION")  :00#POS= 9
myCom.Send("00REA #POSITION")  :00#POS= 19
myCom.Send("00REA #POSITION")  :00#POS= 33
myCom.Send("00REA #POSITION")  :00#POS= 49

Another MOVE_ON command :
myCom.Send("00MOVE_ON -5000")  myCom.Status= 8800
myCom.Status is > 0x8000 ==> an error occured on the last command !!
0x8800: b15 (Fail/Error) = 1  and b11 (Recept Nack or Failed)  = 1
Recept Nack or Failed error is set , use the property DetailCr for error description
myCom.DetailCr  = 0020
0x0020: b05 (Error Cmde ) = 1
doc umac_v6_um_fr.pdf, p29, MOVE_ON command: command reject error if the motor is already in movement !!

wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 8800   00#POS= 962
wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 8800   00#POS= 2922
wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 8800   00#POS= 5906
wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 8800   00#POS= 8539
wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 8800   00#POS= 10001
wait 200ms and try again :
myCom.Send("00MOVE_ON -5000 then 00REA #POS")  Status= 0000   00#POS= 9999

the MICROMAC17 is running from position = 10000 to 5000

Check position  : REA #POSITION , extract µMAC position , until position < 5010
myCom.Send("00REA #POS") 00#POS= 9993 Position =9993
myCom.Send("00REA #POS") 00#POS= 9794 Position =9794
myCom.Send("00REA #POS") 00#POS= 9405 Position =9405
myCom.Send("00REA #POS") 00#POS= 8817 Position =8817
myCom.Send("00REA #POS") 00#POS= 8031 Position =8031
myCom.Send("00REA #POS") 00#POS= 7054 Position =7054
myCom.Send("00REA #POS") 00#POS= 6164 Position =6164
myCom.Send("00REA #POS") 00#POS= 5472 Position =5472
myCom.Send("00REA #POS") 00#POS= 5001 Position =5001


Power OFF = no more motor current
myCom.Send("POWER OFF")

[ESC] = All is done ,quit the program
```

HMS_MiCom2022_MicroMAC_XETAT

- …\MiCom_v2022\Package\examples\MICROMAC17>*HMS_MiCom2022_MicroMac_XEtat.exe*

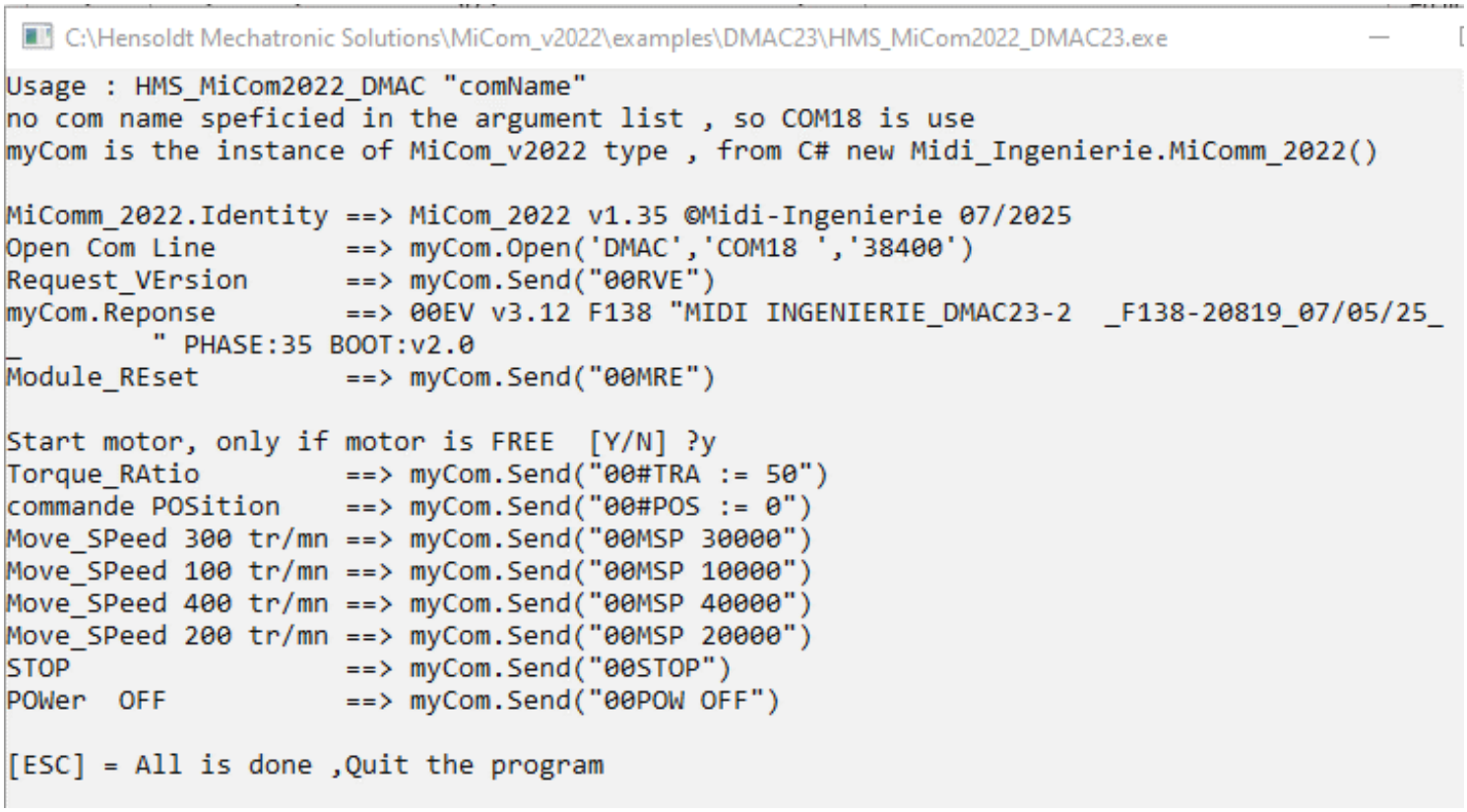
```
C:\Hensoldt Mechatronic Solutions\MiCom_v2022\examples\MICROMAC17\HMS_MiCom2022_MicroMAC_XEtat.exe
Demo_MiCom2022_MicroMAC_XEtat.exe ==> how to use MICom2022 for MICROMAC17, detail on the X_Etat property
Usage : HMS_MiCom2022_MicroMac_XEtat.exe  "ComName"

myCom is the instance of MiCom_v2022 type , from C# new Midi_Ingenierie.MiComm_2022()

Check version and copyright :
MiComm_2022.Identity => MiCom_2022 v1.35 ©Midi-Ingenierie 07/2025

no ComName ref in command line argument : input a ComName >com19
myCom.Open  , select DRIVER , PORTCOM , and BAUDRATE   all as string (C#)
myCom.Open('MICROMAC17', 'com19', '38400') => myCom.Status=0000

myCom.Send  , send a command / requete to the MICROMAC17 driver , adress 00
myCom.Send("00RV") => myCom.Status (or Cr) =0001
Cr=0x0001 , b0=1 : a reponse from the MICROMAC17 exist in myCom.Reponse
myCom.Reponse => 00EV v5.0 9170 "MI_uMAC17_9170-00492_" 0.5

Start motor, only if motor is FREE  [Y/N] ?y
The start command :
myCom.Send("00MOVE_ON +10000") => myCom.Status= 0000

Another MOVE_ON command only when the current movement will be completed  (myCom.X_Etat.b1 = 0)
the X_Etat code is embebed in the protocol trame.
myCom.X_Etat = 83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 534]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 1995]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 4385]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 7218]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 9202]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 10001]   , myCom.X_Etat =81
the last X_Etat.b1=0  (0x81) , so the last movement is completed
myCom.Send("00MOVE_ON -3000") =>  myCom.Status= 0000

Wait for the end of the last MOVE_ON command : check X_Etat
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 9493]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 8087]   , myCom.X_Etat =83
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 7001]   , myCom.X_Etat =81
the last MOVE_ON is ended :

POWER OFF in 2s

myCom.Send("POWER OFF") => myCom.X_Etat =81
!! the X_Etat is set before the final execution of the command
so , with the POWER OFF command , the X_Etat.b0 (power) is still '1'
another command will return the X_Etat.b0 set to 0  which is the POWER OFF command reaction
myCom.Send("00REA #POS") => myCom.Reponse: [00#POS= 7001]   , myCom.X_Etat =80


[ESC] = Quit the program
```

HMS_MiCom2022_MAC23_MAC34



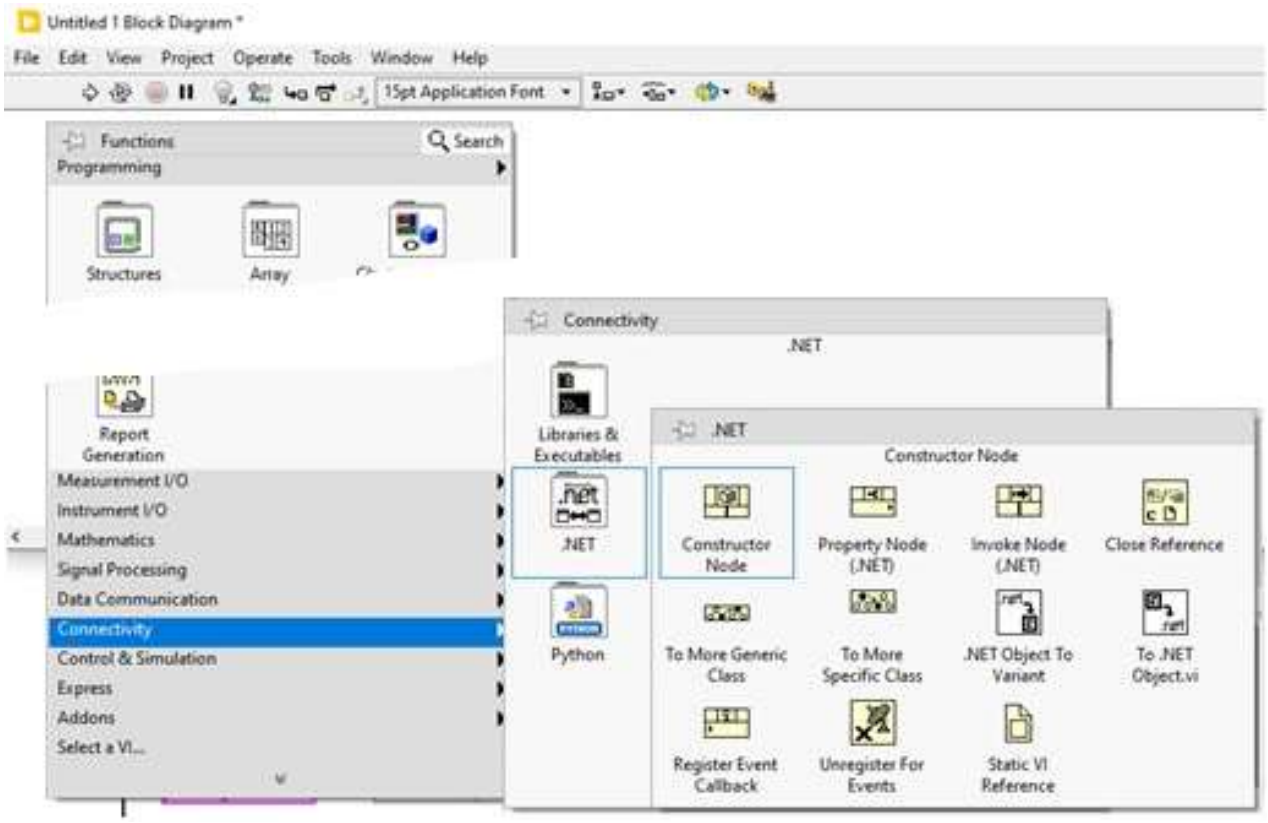
HMS_MiCom2022_MACEXPERT

HMS_MiCom2022_DMAC

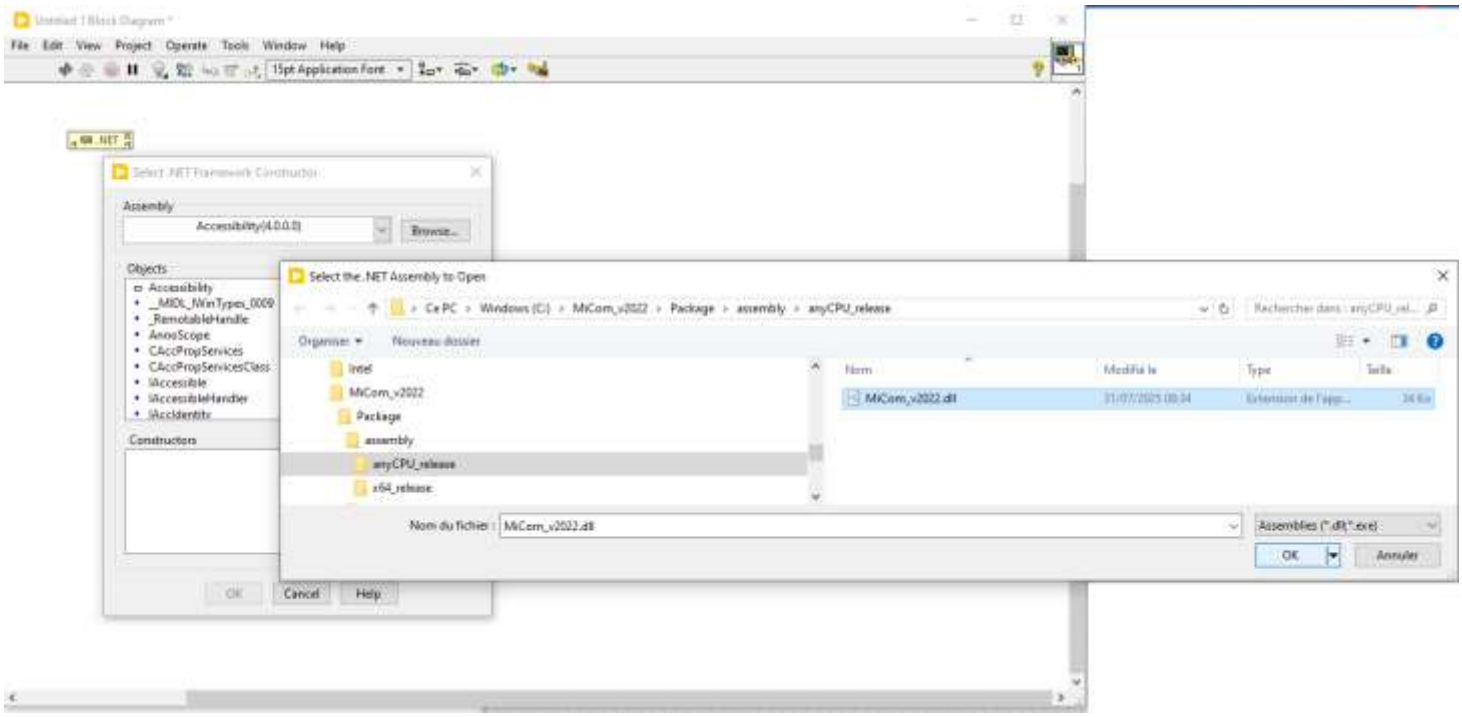


Exemple d'utilisation sous LabView (VI)

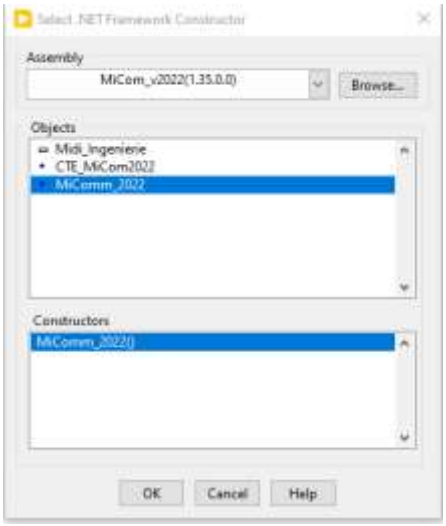
L'intégration dans Labview de l'assembly MiCom_v2022.dll se fait depuis Connectivity directement sur le Block Diagram :



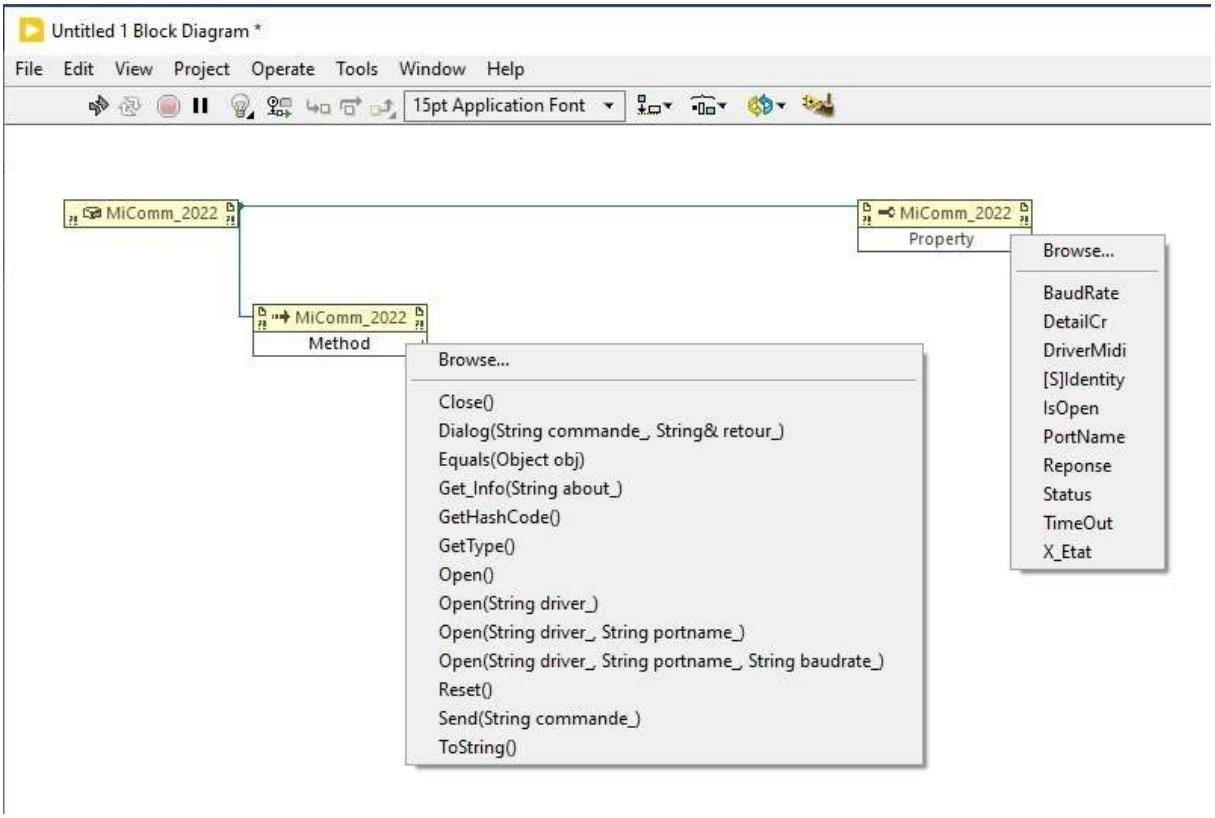
lorsque le Constructor est placé sur le diagram, il faut sélectionner l'assembly associé (MiComv2022.dll) :



En final, il faut sélectionner le constructeur de l'object :



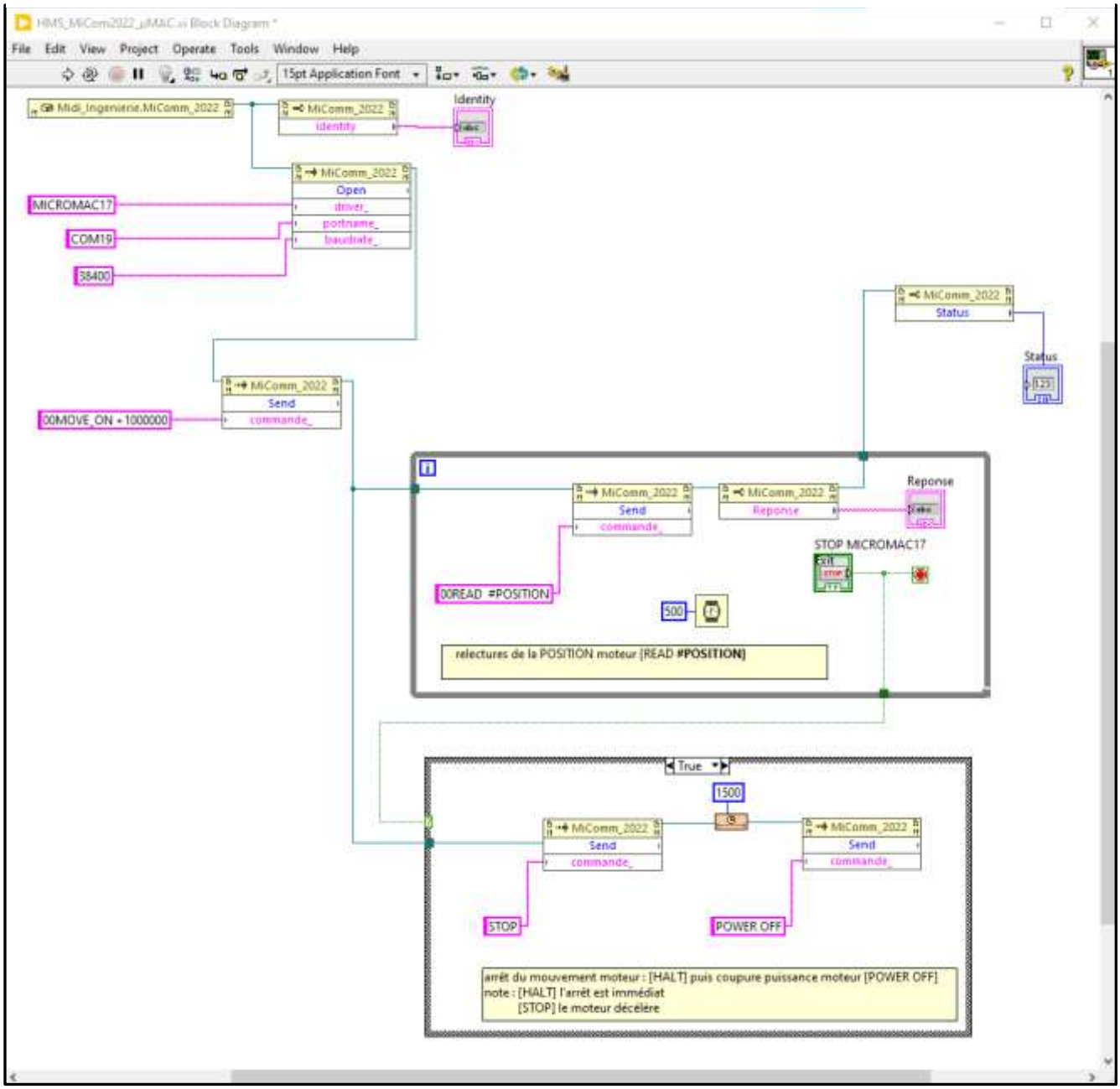
Les properties et methods (invoke) sont accessibles lorsque le constructor est câblé sur le Property node ou le Invoke node :



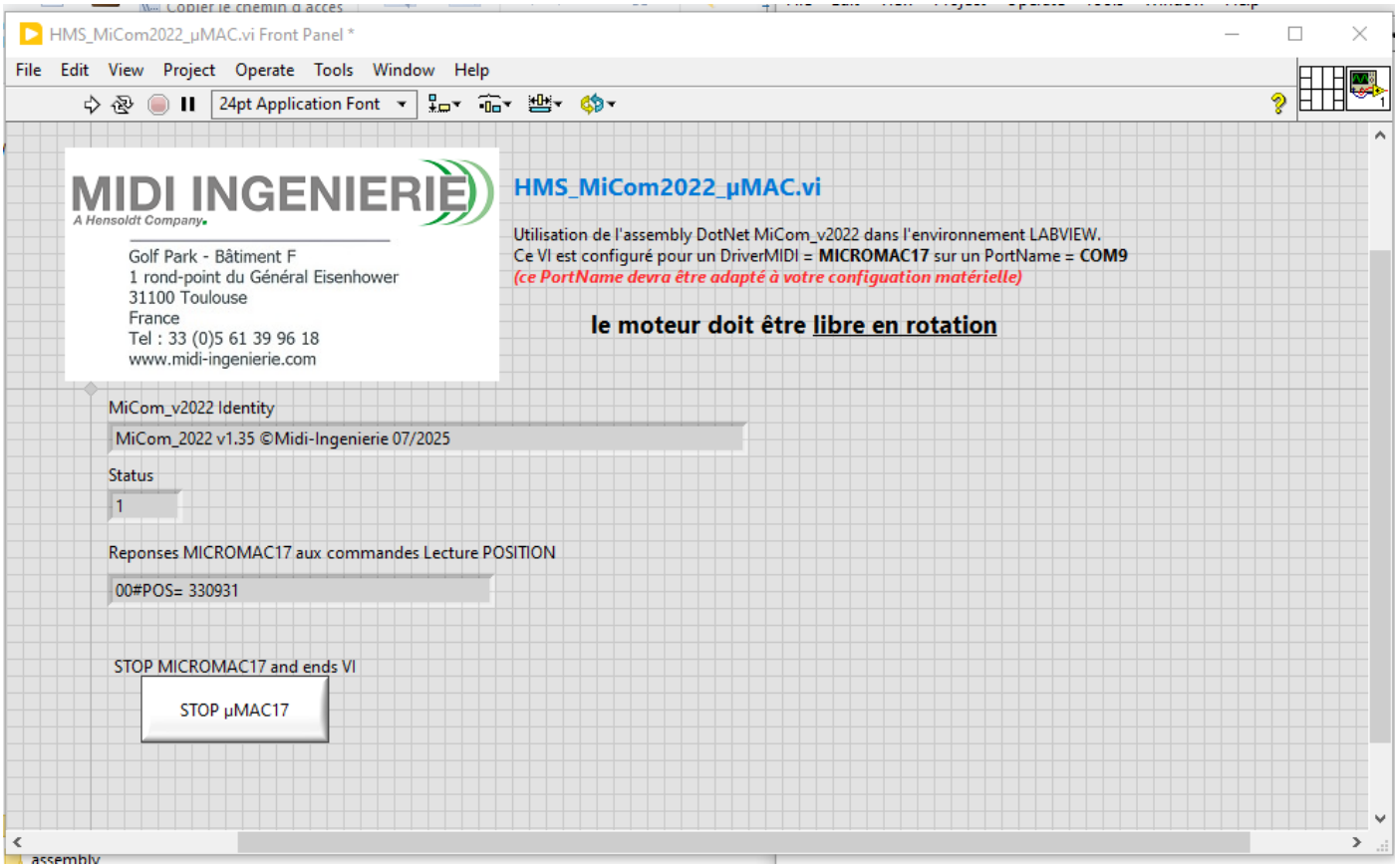
Mise en œuvre d'un mouvement 'long' sur le µMAC , lecture de la position

Commande MOVE_ON , READ #POSITION , STOP , POWER OFF

Block diagram view



Front Panel view after Run and Stop



Définition du Status jusqu'à la version v1.25

			eq. v1.26
	b15	Fail / Error	b15
	b14	baudrate	b05
	b13	portname	b03
	b12	driver_Midi	b04
	b11	close	b07
	b10	open	b08
	b09	not used	b06
	b08	mac	b01
	b07	over_range	b02
	b06	checksum	b10
	b05	frame	b09
	b04	com_lost	b14
	b03	bad reception	b11
	b02	system	b12
	b01	timeout	b13
	b00	response	b00

Mac23, Mac34, MacExpert, MacConverter

Le contrôle des MAC23 et MAC34 se fait exclusivement par le jeu de commande 'Mode Expert'.
https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/mac23_34_ex_v5_um_fr.pdf

Ces commandes sont organisées en index, sous-index, paramètres. Toutes ces données sont à préciser en hexadécimal.
Exemples de commande ModeExpert : (pour un module MAC23 d'adresse 00)

```
00000000000000          // INITIALISATION, page 5 de la documentation
000C000000001A          // LIMITATION DU COUPLE, page 9
000B0000000000          // RAZ POSITION   page 8
```

Rappel : la partie protocole est gérée par MiCom_2022.

Pour contrôler un MAC23 ou un MAC34 en mode EXPERT depuis MiCom_v2022, il faut sélectionner le driver MACEXPERT :

```
//extrait du source HMS_MiCom2022_MACEXPERT.cs
myCom.Open("MACEXPERT",  "COM12" , "38400");
myCom.Send("00000000000000");
myCom.Send("000C000000001A");
myCom.Send("000B0000000000");
```

Le contrôle du MAC23 ou MAC34 depuis MiCom_v2022 permet de disposer d'un jeu de commande STANDARD, moins contraignant que celui du ModeExpert : Ces commandes sont comparables à celles de la famille DMAC, BMAC µMAC.
De plus, ce jeu de commande standard permet d'exploiter directement les requêtes de relectures des paramètres (position, vitesse, status,...)
https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/mac23_v9_um_fr.pdf

Le MacConverter, intégré à MiCom_v2022, est l'outil logiciel qui permet de convertir la commande STANDARD en commande(s) Mode EXPERT, avant émission vers un MAC23/MAC34.

Pour contrôler un MAC23 ou un MAC34 en mode STANDARD depuis MiCom_v2022, il faut sélectionner soit le driver MAC23 ou le driver MAC34 :

```
//extrait du source HMS_MiCom2022_MAC23_MAC34.cs
myCom.Open("MAC23",  "COM12" , "38400");
myCom.Send("00mr") ;
myCom.Send("00gi 20") ;
myCom.Send("00di") ;

myCom.Send("00qv") ;
myCom.Send("00ql") ;
```

L'utilisation du MacConverter est signalée par le bit01 du Status :

0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	MAC34 / MAC23 : nominal hors requête
0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	MAC34 / MAC23 : nominal pour requête

Le MacConverter ne peut convertir que les commandes STANDARD qui ont une équivalence en mode EXPERT :

La commande "etydnssz" ne sera pas convertie, il n'y a donc pas d'émission vers un MAC23. L'erreur est signalée par le bit06 du status (mac converter error)

0x8040	1	0	0	0	0	0		0	0	0	1	0	0	0	0	0	0	erreur conversion MacConverter

Une commande hors limite (ex gi 24510) ne pouvant pas être convertie, sera signalée par le MacConverter Error.
Une commande correctement convertie (ex ga 45000) peut néanmoins générer une erreur d'interprétation par le MAC23 .

la documentation produit

MAC23 https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/mac23_v9_um_fr.pdf

MAC34 https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/mac34_v10_um_fr.pdf

MIBL https://www.midi-ingenierie.com/mibl-documentation/

µMAC https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/umac17_v6_um_fr.pdf

DMAC https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/dmac_v2_um_fr.pdf

BMAC https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/bmac_v2_um_fr.pdf

Mode Expert https://www.midi-ingenierie.com/documentation/ressources/manuels_utilisateurs/mac23_34_ex_v5_um_fr.pdf