Librairie logicielle pour la gestion de base de données

DBX-MUX-DLL

Guide utilisateur

juil.-05

CV/061354-02

TABLE DES MATIERES

1 But du document

1.1 But

Le but de ce document est de décrire les différentes fonctions composant les bibliothèques d'accès aux bases de données. Cette bibliothèque de fonctions logicielles permettent d'interfacer une application PC à des bases de données *.dbc, *.dbv, *.dbk, *.dbn, *.dbl.

L'utilisation des ces fonctions permet à l'utilisateur de rendre transparent l'application par rapport à la définition des messages et des signaux.

Toutes ces fonctions sont regroupées au sein d'une librairie dynamique (DLL), cette librairie est opérationnelle pour des PC équipés de systèmes d'exploitation Windows 95/98, Windows NT, 2000 et XP.

2 Chargement de la librairie

Il existe deux méthodes pour inclure les DLLs au projet :
- La méthode dite par chargement statique
- La méthode dite par chargement dynamique

2.1 Méthode par chargement statique

Cette méthode consiste à appeler directement les fonctions de la librairie en référence externe, comme n'importe quelles fonctions standard, puis au moment de l'édition de lien d'insérer le fichier *.LIB

2.1.1 Procédure à suivre

1 – Insérer dans chaque fichier source faisant appel à au moins une fonction des librairies le fichier de définition des prototype DbxDll.h.

#include " DbxDll.h "

2 – Ajouter au projet le fichier DbxDll.LIB

Remarques :
- Avec VISUAL C++ et BORLAND C++, cette opération se réalise par le menu « Projet » puis « ajouter au projet »
- Avec LabWindows CVI :
- Ajouter à votre projet le fichier DbxDll.h
- Editer ce fichier, puis sélectionner le menu « Options » / « Generate Dll import Library …»
- Sélectionner la librairie : DbxDll.dll
- Ajouter à votre projet le fichier DbxDll.lib

2.1.2 Avantages et inconvénients

Avantages :
1. Méthode très simple à mettre en œuvre

Inconvénients :
1. Il se peut qu'en fonction de certains compilateurs utilisés, le fichier .LIB ne soit pas compatible. Dans ce cas une erreur de format est produite à l'utilisateur et seule la méthode de chargement dynamique est possible.

2. Une fois l'application générée, l'application est dépendante de la version du fichier DbxDll.dll. Une mise à jour de la librairie implique une re-compilation de votre application.

2.2 Méthode par chargement dynamique

Cette méthode consiste à appeler indirectement les fonctions de la librairie. Cela se réalise par un premier appel à la fonction **DBX_LoadDLL** qui a pour rôle d'initialiser des pointeurs de fonctions vers les librairies. Par la suite, l'appel aux fonctions a lieu comme si celles-ci étaient directement liées au projet.

Remarque :

2.2.1 Procédure à suivre

1 - Insérer dans votre fichier source le fichier de définition des prototypes DbxDll.h avec au préalable le mot clef DBX_DYNAMIC_DLL indiquant le chargement dynamique des fonctions.

```
#define DBX_DYNAMIC_DLL
#include " DbxDll.h"
```

2 – Appeler la fonction DBX_LoadDLL avant tout autre appel de fonction

3 – Insérer dans les autres fichiers faisant éventuellement appel aux librairies le fichier de définition DbxDll.h avec le mot clef DBX_DYNAMIC_DLL (chargement dynamique) et DBX_DYNAMIC_DLL_EXTERN pour ne pas re-déclarer les pointeurs de fonctions.

```
#define DBX_DYNAMIC_DLL
#define DBX_DYNAMIC_DLL_EXTERN
#include " DbxDll.h"
```

4 – Appeler la fonction DBX_FreeDLL en fin de programme pour libérer les pointeurs de fonctions

2.2.2 Avantages et inconvénients

Avantages :

1. L'application accède aux fonctions par l'intermédiaire de requêtes windows (API WIN32), l'application peut se compiler sans problème de format de fichier avec la librairie.
2. L'application n'est plus dépendante du fichier DbxDll.dll, une mise à jour uniquement des DLLs peut être envisagées sans re-compiler l'application.

Inconvénients :

1. Méthode légèrement plus compliquée à mettre en œuvre que la méthode par chargement statique

3 Bibliothèques communes à tous les types de base de données

3.1 DBX_LoadDLL: Chargement dynamique de la librairie (Accès direct)

Prototype :
tDbxStatus DBX_LoadDLL(char * szSpecPath);

Description :
Cette fonction est nécessaire lorsque l'application désire charger les fonctions dynamiquement.
Cette fonction est la première fonction que doit appeler l'application, cet appel est unique et se situe en début de programme.

exemple :
```
if (DBX_LoadDLL () != DBX_STATUS_OK)
{ /* Une erreur s'est produite */
        printf("ERREUR chargement librairie");
}
```

Paramètres d'entrées :
Aucun

Paramètres de sorties :
Aucun

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_FILEDLL	Erreur de chargement de la DLL
DBX_STATUS_ERR_OPENDLL	Erreur d'initialisation de la DLL

3.2 DBX_FreeDLL: Libération du chargement dynamique de la librairie

Prototype :
tMuxStatus DBX_FreeDLL (void);

Description :
Cette fonction est nécessaire lorsque l'application désire charger les fonctions dynamiquement.
Cette fonction est la dernière fonction que doit appeler l'application.

exemple :

```
if (DBX_FreeDLL () != DBX_STATUS_OK)
{ /* Une erreur s'est produite */
        printf("ERREUR");
}
```

Paramètres d'entrées :
Aucun

Paramètres de sorties :
Aucun

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_FILEDLL	Erreur lors du déchargement de la DLL

3.3 DBX_OpenDatabase : Ouvre une base de données

Prototype :
tDbxStatus DBX_OpenDatabase(char *szDatabase, tDataBaseHandle * hDataBaseHandle);

Description :
Cette fonction charge une base de données.

Paramètres d'entrées :
szDatabase : Chemin de la base de données : « C:\\MaBase.dbc »

Paramètres de sorties :
hDataBaseHandle: Handle sur la base de données

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK
DBX_STATUS_ERR_OPEN Erreur d'ouverture de la base

3.4 DBX_CloseDatabase: Ferme une base de données

Prototype :
tDbxStatus DBX_CloseDatabase(tDataBaseHandle hDataBaseHandle);

Description :
Cette fonction ferme une base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données.

Paramètres de sorties :
Aucun.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK
DBX_STATUS_ERR_CLOSE Erreur de fermeture de la base

4 Fonctions CAN

4.1 DBX_GetCANMsg : Retourne la définition d'un message

Prototype :
tDbxStatus DBX_GetCANMsg(tDataBaseHandle hDataBaseHandle, DWORD dwIdent, tCanMsg* pstCanMessage);

Description :
Cette fonction retourne la définition d'un message.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
dwIdent: Identifiant du message.

Paramètres de sorties :
pstCanMessage: Message mis à jour à partir des informations contenues dans la base de données.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_MSGNOTFOUND	Message non trouvé dans la base de données

4.2 DBX_GetCANSignalPhysVal: Retourne la valeur physique d'un signal

Prototype :
tDbxStatus DBX_GetCANSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tCanEvent * hCanEvent, char * szSignalName, float * fValue) ;

Description :
Cette fonction retourne la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
hCanEvent: Message contenant les données à décoder.

szSignalName : Nom du signal à décoder.

Paramètres de sorties :
fValue: Valeur physique du signal.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

4.3 DBX_SetCANSignalPhysVal: Met à jour la valeur physique d'un signal

Prototype :
tDbxStatus DBX_SetCANSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tCanMsg * hCanMsg, char * szSignalName, float fValue)

Description :
Cette fonction met à jour la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
szSignalName : Nom du signal à mettre à jour.
fValue: Nouvelle valeur physique du signal.

Paramètres de sorties :
hCanMsg: Message contenant les données à mettre à jour.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

5 Fonctions VAN

5.1 DBX_GetVANMsg : Retourne la définition d'un message

Prototype :
tDbxStatus DBX_GetVANMsg(tDataBaseHandle hDataBaseHandle, DWORD dwIdent, tVanMsg* hVanMessage) ;

Description :
Cette fonction retourne la définition d'un message.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
dwIdent: Identifiant du message.

Paramètres de sorties :
hVanMessage: Message mis à jour à partir des informations contenues dans la base de données.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_MSGNOTFOUND	Message non trouvé dans la base de données

5.2 DBX_GetVANSignalPhysVal: Retourne la valeur physique d'un signal

Prototype :
tDbxStatus DBX_GetVANSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tVanEvent * hVanEvent, char * szSignalName, float * fValue) ;

Description :
Cette fonction retourne la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
hVanEvent: Message contenant les données à décoder.

szSignalName : Nom du signal à décoder.

Paramètres de sorties :
fValue: Valeur physique du signal.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

5.3 DBX_SetVANSignalPhysVal: Met à jour la valeur physique d'un signal

Prototype :
tDbxStatus DBX_SetVANSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tVanMsg * hVanMsg, char * szSignalName, float fValue)

Description :
Cette fonction met à jour la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
szSignalName : Nom du signal à mettre à jour.
fValue: Nouvelle valeur physique du signal.

Paramètres de sorties :
hVanMsg: Message contenant les données à mettre à jour.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

6 Fonctions LIN

6.1 DBX_GetLINMsg : Retourne la définition d'un message

Prototype :
tDbxStatus DBX_GetLINMsg(tDataBaseHandle hDataBaseHandle, DWORD dwIdent, tLinMsg* hLinMessage);

Description :
Cette fonction retourne la définition d'un message.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
dwIdent: Identifiant du message (6 bits).

Paramètres de sorties :
hLinMessage: Message mis à jour à partir des informations contenues dans la base de données.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_MSGNOTFOUND	Message non trouvé dans la base de données

6.2 DBX_GetLINSignalPhysVal: Retourne la valeur physique d'un signal

Prototype :
tDbxStatus DBX_GetLINSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tLinEvent * hLinEvent, char * szSignalName, float * fValue);

Description :
Cette fonction retourne la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
hLinEvent: Message contenant les données à décoder.

szSignalName : Nom du signal à décoder.

Paramètres de sorties :
fValue: Valeur physique du signal.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

6.3 DBX_SetLINSignalPhysVal: Met à jour la valeur physique d'un signal

Prototype :
tDbxStatus DBX_SetLINSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tLinMsg * hLinMsg, char * szSignalName, float fValue) ;

Description :
Cette fonction met à jour la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
szSignalName : Nom du signal à mettre à jour.
fValue: Nouvelle valeur physique du signal.

Paramètres de sorties :
hLinMsg: Message contenant les données à mettre à jour.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

7 Fonctions ISO14230

7.1 DBX_GetISO14230Msg: Retourne la définition d'un message

Prototype :
```
tDbxStatus DBX_GetISO14230Msg(tDataBaseHandle hDataBaseHandle,
  unsigned short  wSrcAddr, unsigned short  wDstAddr, unsigned short  wService,
  unsigned short  wSubService, tIso14230Msg* hIso14230Message) ;
```

Description :
Cette fonction retourne la définition d'un message.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
wSrcAddr: code adresse de l'émetteur du message.
wDstAddr: code adresse du récepteur du message.
wService: code du service ISO14230.
wSubService: code du sous service ISO14230.

Paramètres de sorties :
hIso14230Message: Message mis à jour à partir des informations contenues dans la base de données.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_MSGNOTFOUND	Message non trouvé dans la base de données

7.2 DBX_GetISO14230SignalPhysVal: Retourne la valeur physique d'un signal

Prototype :
```
tDbxStatus DBX_GetISO14230SignalPhysVal_Float(tDataBaseHandle hDataBaseHandle,
  tIsoEvent * hIsoEvent, char * szSignalName, float * fValue) ;
```

Description :
Cette fonction retourne la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
hIsoEvent: Message contenant les données à décoder.
szSignalName : Nom du signal à décoder.

Paramètres de sorties :
fValue: Valeur physique du signal.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

7.3 DBX_SetISO14230SignalPhysVal: Met à jour la valeur physique d'un signal

Prototype :
tDbxStatus DBX_SetISO14230SignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tIso14230Msg * hIso14230Message, char * szSignalName, float fValue);

Description :
Cette fonction met à jour la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
szSignalName : Nom du signal à mettre à jour.
fValue: Nouvelle valeur physique du signal.

Paramètres de sorties :
hIso14230Message: Message contenant les données à mettre à jour.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

8 Fonctions NWC

8.1 DBX_GetNWCMsg: Retourne la définition d'un message

Prototype :
tDbxStatus DBX_GetNWCMsg(tDataBaseHandle hDataBaseHandle, DWORD dwIdent, unsigned short wService, unsigned short wSubService, tNwcMsg* hNwcMessage);

Description :
Cette fonction retourne la définition d'un message.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
dwIdent : Identifiant du message.
wService: code du service de diagnostic.
wSubService: code du sous service de diagnostic.

Paramètres de sorties :
hNwcMessage: Message mis à jour à partir des informations contenues dans la base de données.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_MSGNOTFOUND	Message non trouvé dans la base de données

8.2 DBX_GetNWCSignalPhysVal: Retourne la valeur physique d'un signal

Prototype :
tDbxStatus DBX_GetNWCSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, tNwcEvent * hNwcEvent, char * szSignalName, float * fValue);

Description :
Cette fonction retourne la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :

hDataBaseHandle: Handle sur la base de données
hNwcEvent: Message contenant les données à décoder.
szSignalName : Nom du signal à décoder.

Paramètres de sorties :
fValue: Valeur physique du signal.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

8.3 DBX_SetNWCSignalPhysVal: Met à jour la valeur physique d'un signal

Prototype :
tDbxStatus DBX_SetNWCSignalPhysVal_Float(tDataBaseHandle hDataBaseHandle, DWORD dwIdent, tNwcMsg * hNwcMessage, char * szSignalName, float fValue);

Description :
Cette fonction met à jour la valeur physique d'un signal à partir des informations contenues dans la base de données.

Paramètres d'entrées :
hDataBaseHandle: Handle sur la base de données
szSignalName : Nom du signal à mettre à jour.
fValue: Nouvelle valeur physique du signal.

Paramètres de sorties :
hNwcMessage: Message contenant les données à mettre à jour.

Code retour :
Status : Compte rendu d'exécution de la fonction

DBX_STATUS_OK	
DBX_STATUS_ERR_INVALIDPARAM	Erreur de paramètres
DBX_STATUS_ERR_SGLNOTFOUND	Signal non trouvé dans la base de données
DBX_STATUS_ERR_INVALIDSGL	Définition du signal incohérente

Annexe : Prototypes communs

```
tDbxStatus  DBX_LoadDLL(char * szSpecPath);
tDbxStatus  DBX_FreeDLL(void);
tDbxStatus  _DLLEXP DBX_OpenDatabase(char *szDatabase, tDataBaseHandle *
hDataBaseHandle);
tDbxStatus  _DLLEXP DBX_CloseDatabase(tDataBaseHandle hDataBaseHandle);
```

Annexe : Prototypes CAN

```
tDbxStatus _DLLEXP DBX_GetCANMsg(tDataBaseHandle hDataBaseHandle,
  DWORD dwIdent, tCanMsg* pstCanMessage);
tDbxStatus _DLLEXP DBX_GetCANSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tCanEvent * hCanEvent, char * szSignalName, float * fValue);
tDbxStatus _DLLEXP DBX_SetCANSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tCanMsg * hCanMsg, char * szSignalName, float fValue);
```

Annexe : Prototypes VAN

```
tDbxStatus _DLLEXP DBX_GetVANMsg(tDataBaseHandle hDataBaseHandle,
  DWORD dwIdent, tVanMsg* hVanMessage);
tDbxStatus _DLLEXP DBX_GetVANSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tVanEvent * hVanEvent, char * szSignalName, float * fValue);
tDbxStatus _DLLEXP DBX_SetVANSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tVanMsg * hVanMsg, char * szSignalName, float fValue);
```

Annexe : Prototypes LIN

```
tDbxStatus _DLLEXP DBX_GetLINMsg(tDataBaseHandle hDataBaseHandle,
  DWORD dwIdent, tLinMsg* hLinMessage);
tDbxStatus _DLLEXP DBX_GetLINSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tLinEvent * hLinEvent, char * szSignalName, float * fValue);
tDbxStatus _DLLEXP DBX_SetLINSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tLinMsg * hLinMsg, char * szSignalName, float fValue);
```

Annexe : Prototypes ISO14230

```
tDbxStatus _DLLEXP DBX_GetISO14230Msg(tDataBaseHandle hDataBaseHandle,
  unsigned short  wSrcAddr, unsigned short  wDstAddr, unsigned short  wService,
  unsigned short  wSubService, tIso14230Msg* hIso14230Message);
tDbxStatus _DLLEXP DBX_GetISO14230SignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tIsoEvent * hIsoEvent, char * szSignalName, float * fValue);
tDbxStatus _DLLEXP DBX_SetISO14230SignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tIso14230Msg * hIso14230Message, char * szSignalName, float
  fValue);
```

Annexe : Prototypes NWC

```
tDbxStatus _DLLEXP DBX_GetNWCMsg(tDataBaseHandle hDataBaseHandle,
  DWORD dwIdent, unsigned short  wService, unsigned short  wSubService,
  tNwcMsg* hNwcMessage);
tDbxStatus _DLLEXP DBX_GetNWCSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, tNwcEvent * hNwcEvent, char * szSignalName, float * fValue);
tDbxStatus _DLLEXP DBX_SetNWCSignalPhysVal_Float(tDataBaseHandle
  hDataBaseHandle, DWORD dwIdent, tNwcMsg * hNwcMessage, char *
  szSignalName, float fValue);
```

Liste des éditions successives

Version	Date	Auteur	Modifications
01	06/2005	C.VOLLAT	Création du document
02	07/2005	C.VOLLAT	Ajustement de la conversion des entiers Modification du type du paramètre fValue, utilisé pour mettre à jour la valeur d'un signal