mechapro

Dokumentation zur DLL Sendeframe_DLL

NC-Pilot USB

Stand: 09.01.2009

Structs / Stukturen:

```
// in WINBASE.H:
typedef struct _OVERLAPPED {
    DWORD   Internal;
    DWORD   InternalHigh;
    DWORD   Offset;
    DWORD   OffsetHigh;
    HANDLE  hEvent;
} OVERLAPPED, *LPOVERLAPPED;

typedef struct TagCommSet
{
        HANDLE              hCommDev;
        char                DeviceName[5];
        char                controllername[16];
        unsigned long       ulBaudRate;
        unsigned char       FrameCount;     //      Frame Counter 0 bis 255.
        unsigned char       ErrorCode;
        unsigned char       ucByteSize;
        unsigned char       ucParity;
        unsigned char       ucStopBits;
        unsigned char       ucFlowCtrl;
        OVERLAPPED          osWrite;
        OVERLAPPED          osRead;
} NCStepCommPortSettings;
```

Umsetzung in Visual Basic:

```
Public Type test
  Handle As Long
  DeviceName(4) As Byte
  controllername(15) As Byte
  ulBaudRate As Long
  FrameCount As Byte
  ErrorCode As Byte
  ucByteSize As Byte
  ucParity As Byte
  ucStopBits As Byte
  ucFlowCtrl As Byte
  osWrite As OVERLAPPED
  osRead As OVERLAPPED
End Type
```

Der Typ OVERLAPPED ist in der WINBASE.H definiert:

```
Public Type OVERLAPPED
   iternal As Long
   iternalHigh As Long
   offset As Long
   OffsetHigh As Long
   hEvent As Long
End Type
```

Funktionen

Rückgabewert für alle Funtkionen:

// 0 - failed
// 1 - success

Interne Funktionen:

int Set_CommPortSettings(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned int PortNo);

int NCStepOpenConnection(NCStepCommPortSettings *pNCStepCommPortSettings);

int NCStepWriteCommBlock(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char *sf, unsigned long ulBytesToWrite);

int NCStepReadCommBlock(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char *af, unsigned long ulBytesToRead);

Externe Funktionen:

int NCStepWriteChar(NCStepCommPortSettings *pNCStepCommPortSettings, char *c_sf, unsigned long ulBytesToWrite);

Wird nur für Firmwareupdate verwendet

int NCStepReadChar(NCStepCommPortSettings *pNCStepCommPortSettings, char *c_af, unsigned long ulBytesToRead);

Wird nur für Firmwareupdate verwendet

LIBSPEC int __stdcall NCStepOpenConnectionPort(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned int PortNo);

Öffenen eines definierten COM-Ports. Übergabe der COM-Port Settings (siehe Header-Datei).

LIBSPEC int __stdcall NCStepOpenConnectionAuto(NCStepCommPortSettings *pNCStepCommPortSettings);

Sucht die COM-Ports 1-8 nach einem Controller ab und öffnet den ersten Port, an dem ein NC-Pilot gefunden wird. Die COM-Port Nummer wird in die Struktur NCStepCommPortSettings eingetragen.

LIBSPEC int __stdcall NCStepOpenConnectionName(NCStepCommPortSettings *pNCStepCommPortSettings, char *controllername);

Sucht die COM-Ports 1-8 nach einem Controller mit dem angegebenen Namen ab und öffnet den ersten Port, an dem ein NC-Pilot mit dem passenden Namen gefunden wird. Die COM-Port Nummer wird in die Struktur NCStepCommPortSettings eingetragen.

LIBSPEC int __stdcall NCStepCloseConnection(NCStepCommPortSettings *pNCStepCommPortSettings);

Schließt eine bestehende Verbindung.

LIBSPEC int __stdcall FahreRelativ_XY_Schritte(NCStepCommPortSettings *pNCStepCommPortSettings, long BahnV, unsigned char AnzKoordPaare, long *x, long *y, unsigned char rampenbits);

LIBSPEC int __stdcall FahreRelativ_ZC_Schritte(NCStepCommPortSettings *pNCStepCommPortSettings, long BahnV, unsigned char AnzKoordPaare, long *z, long *c, unsigned char rampenbits);

Kommando für relative Verfahrbewegung in Schritten
Parameter:
BahnV - Bahngeschwindigkeit, long. Zum Verwenden des Default-Wertes 0 eintragen.
Default ist die in Menü „Einstellungen“ - „Geschwindigkeiten“->„Fahrgeschwindigkeit“ eingestellte Vorschubgeschwindigkeit.
AnzKoordPaare - Anzahl der im Befehl übergebenen Koordinatenpaare (1 bis 6)
x bzw z - Array mit 6 Elementen von Typ „long“
y bzw c - Array mit 6 Elementen von Typ „long“
rampenbits - Siehe Bedienungsanleitung „NC-Pilot USB“, legt fest, an welchen Stellen der Bewegung (Koordinatenpaare) mit Bremsrampe gefahren werden soll. Die Auswertung erfolgt bitweise

LIBSPEC int __stdcall FahreRelativ_XY_mm(NCStepCommPortSettings *pNCStepCommPortSettings, float BahnV, unsigned char AnzKoordPaare, float *x, float *y, unsigned char rampenbits);

LIBSPEC int __stdcall FahreRelativ_ZC_mm(NCStepCommPortSettings *pNCStepCommPortSettings, float BahnV, unsigned char AnzKoordPaare, float *z, float *c, unsigned char rampenbits);

Parameter wie oben, außer:
BahnV - Bahngeschwindigkeit, float.
x bzw z - Array mit 6 Elementen von Typ „float“
y bzw c - Array mit 6 Elementen von Typ „float“

LIBSPEC int __stdcall NullPunktSetzen_Schritte(NCStepCommPortSettings *pNCStepCommPortSettings, long x, long y, long z, long c);

Parameter:
x - Nullpunkt X-Achse, long
y - Nullpunkt Y-Achse, long
z - Nullpunkt Z-Achse, long
c - Nullpunkt C-Achse, long

LIBSPEC int __stdcall NullPunktSetzen_mm(NCStepCommPortSettings *pNCStepCommPortSettings, float x, float y, float z, float c);

Parameter:
x - Nullpunkt X-Achse, float
y - Nullpunkt Y-Achse, float
z - Nullpunkt Z-Achse, float
c - Nullpunkt C-Achse, float

LIBSPEC int __stdcall Referenzfahrt(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Anz_Achs, unsigned char *Nummer);

Ändern der Reihenfolge der referenzierten Achsen. Die Richtung der Achsen bleibt so, wie in sie in den Einstellungen definiert ist.
Parameter:

Anz_Achs	- Anzahl der Achsen (0 <= No_Achs <= 4).
Anz_Achs = 0	- alle 4 Achsen werden referenziert. Die Richtung und Reihenfolge sind im diesem Fall aus Menü „Einstellungen“ - „Mechanik“ -> „Richtung/Sequenz Referenzfahrt“ zu übernehmen.
Anz_Achs = 1,2,3,4	- Die angegebene Anzahl Achsen werden referenziert.
Nummer[0]	- Nummer der ersten Achse (0=X, 1=Y, 2=Z, 3=C)
Nummer[1]	- Nummer der zweiten Achse (0=X, 1=Y, 2=Z, 3=C)
Nummer[2]	- Nummer der dritten Achse (0=X, 1=Y, 2=Z, 3=C)
Nummer[3]	- Nummer der vierten Achse (0=X, 1=Y, 2=Z, 3=C)

LIBSPEC int __stdcall ParkPositionAnfahren(NCStepCommPortSettings *pNCStepCommPortSettings);

Kommando, um die in den Parametern hinterlegte Parkposition anzufahren.

LIBSPEC int __stdcall SpindelKuhlungSetzen(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char sp, unsigned char kuh);

Schalten der Ausgänge „Spindel“ und „Kühlung“
Parameter:
sp - Spindel 0=aus, 1=ein
kuh - Kühlung 0=aus, 1=ein

LIBSPEC int __stdcall AllgemeineAusgaeneSetzen(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char output3);

Parameter:
output3 - Byte, das auf den Zusatzausgängen 0 bis 7 ausgegeben wird (0=aus, 1=ein)

LIBSPEC int __stdcall NotStop_ohneBremsrampe(NCStepCommPortSettings *pNCStepCommPortSettings);

Sofortiger Stopp der Verfahrbewegung (ohne Bremsrampe). Der Inhalt des Befehlspuffers wird verworfen. Nach dem Befehl muss die Funktion„FehlerQuittieren“ ausgeführt werden, da es zu Schrittverlusten kommen kann. Nach einem Notstopp sollte eine Referenzfahrt durchgeführt werden.

LIBSPEC int __stdcall NotHalt_mitBremsrampe(NCStepCommPortSettings *pNCStepCommPortSettings);

Sofortiger Stopp der Verfahrbewegung (mit Bremsrampe). Der Inhalt des Befehlspuffers wird verworfen. Die Achspositionen bleiben gültig.

LIBSPEC int __stdcall StatusAbfrage(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char basiseinheit);

Zur Abfrage von Fehlerzuständen die Funktionen GetPosition_Schritte bzw. GetPosition_mm verwenden!
Parameter:
basiseinheit - Basiseinheit im Antwortframe (0-mm, 1-Schritte).

LIBSPEC int STDCALL GetPosition_Schritte(NCStepCommPortSettings *pNCStepCommPortSettings, long *ist_steps, unsigned char* Error);

Parameter:
ist_steps - Array mit 4 Elementen von Typ „long“ (Istpositionen X, Y, Z, C)
Error - Fehlercode des Controllers.
Mögliche Fehlercodes:

0x00: kein Fehler
0x01: Befehlspuffer nicht leer (nur bei Betriebsartwechsel)
0x02: Achse(n) fahren noch (nur bei Betriebsartwechsel)
0x03: Menüpunkt/Befehl unbekannt (nur bei Betriebsartwechsel)
0x11: X-Achse untere Grenze erreicht
0x12: X-Achse obere Grenze erreicht
0x13: Endschalter X-Achse betätigt
0x21: Y-Achse untere Grenze erreicht
0x22: Y-Achse obere Grenze erreicht
0x23: Endschalter X-Achse betätigt
0x31: Z-Achse untere Grenze erreicht
0x32: Z-Achse obere Grenze erreicht
0x33: Endschalter X-Achse betätigt
0x41: C-Achse untere Grenze erreicht
0x42: C-Achse obere Grenze erreicht
0x43: Endschalter X-Achse betätigt
0x53: Notausschalter betätigt
0x71: Parameter unzulässig

0x91: Ungültige Parameternummer
0x99: Notstopp ausgelöst, Referenzfahrt erforderlich
0x0A: Anzahl der Achsen für Referenzfahrt falsch

LIBSPEC int STDCALL GetPosition_mm(NCStepCommPortSettings *pNCStepCommPortSettings, float *ist_mm, unsigned char* Error);

Parameter:
ist_steps - Array mit 4 Elementen von Typ „float“ (Istpositionen X, Y, Z, C)
Error - Fehlercode des Controllers (siehe Funktion GetPosition_Schritte)

LIBSPEC int __stdcall BetriebsArtenWechsel(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char ziel_ba);

Wechsel der Betriebsart (Autmatikmodus nach Parameterübertragung oder umgekehrt).
Parameter:
ziel_ba - 0x01:Parameterübertragung, 0x02:Automatikbetrieb

LIBSPEC int __stdcall FehlerQuitieren(NCStepCommPortSettings *pNCStepCommPortSettings);

Quittieren eine Fehlermeldung des Controllers.

LIBSPEC int __stdcall ParameterAendern_long(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Hbyte, unsigned char Lbyte, long param);

Ändern eines Parameters im Format long.
Auf die Funktionen „ParameterAendern_long“, „ParameterAendern_4_longs“
„ParameterAendern_float“ und „ParameterAendern_Richtung_Sequenz_ReferenzFahrt“
wird mit dem normalen Antwortframe geantwortet (s.Tabele 2 der „Bedienungsanleitung“)
Parameter:
Hbyte - Parameternummer highbyte aus den #define HBYTE_...
Lbyte - Parameternummer lowbyte aus den #define LBYTE_...
param - Parameterwert von Typ „long“, wird über Hbyte und Lbyte adressiert

LIBSPEC int __stdcall ParameterAendern_4_longs(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Hbyte, unsigned char Lbyte, long paramX, long paramY, long paramZ, long paramC);

Ändern eines Parameterssatzes im Format long[4].
Parameter:
Hbyte - Parameternummer highbyte aus den #define HBYTE_...
Lbyte - Parameternummer lowbyte aus den #define LBYTE_...
paramX - Parameterwert von Typ „long“, wird über Hbyte und Lbyte adressiert
paramY - Parameterwert von Typ „long“, wird über Hbyte und Lbyte adressiert
paramZ - Parameterwert von Typ „long“, wird über Hbyte und Lbyte adressiert
paramC - Parameterwert von Typ „long“, wird über Hbyte und Lbyte adressiert

LIBSPEC int __stdcall ParameterAendern_4_uchar(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Hbyte, unsigned char Lbyte, unsigned char paramX, unsigned char paramY, unsigned char paramZ, unsigned char paramC);

Ändern eines Parameterssatzes im Format unsigned char[4].
Parameter:
Hbyte - Parameternummer highbyte aus den #define HBYTE_...
Lbyte - Parameternummer lowbyte aus den #define LBYTE_...
paramX - Parameterwert von Typ „unsigned char“, wird über Hbyte und Lbyte adressiert
paramY - Parameterwert von Typ „unsigned char“, wird über Hbyte und Lbyte adressiert
paramZ - Parameterwert von Typ „unsigned char“, wird über Hbyte und Lbyte adressiert
paramC - Parameterwert von Typ „unsigned char“, wird über Hbyte und Lbyte adressiert

LIBSPEC int __stdcall ParameterAendern_float(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Hbyte, unsigned char Lbyte, long param, unsigned char *sf, unsigned char *af);

Ändern eines Parameters im Format float.
Parameter:

Hbyte	- Parameternummer highbyte aus den #define HBYTE_...
Lbyte	- Parameternummer lowbyte aus den #define LBYTE_...
param	- Parameterwert von Typ „float“, wird über Hbyte und Lbyte adressiert

LIBSPEC int __stdcall ParameterAendern_Richtung_Sequenz_ReferenzFahrt(NCStepCommPortSettings *pNCStep-CommPortSettings, unsigned char Hbyte, unsigned char Lbyte, unsigned char richtung_achse1, unsigned char richtung_achse2, unsigned char richtung_achse3, unsigned char richtung_achse4);

Legt Richtung und Reihenfolge für die Referenzfahrt fest.
Parameter:

Hbyte	- aus den #define HBYTE_...
Lbyte	- aus den #define LBYTE_...
richtung_achse1	- Richtung in der für die erste zu referenzierenden Achse nach dem Schalter gesucht wird (0-positive, 1-negative, 2-Achse wird nicht referenziert)
richtung_achse2	- Richtung in der für die zweite zu referenzierenden Achse nach dem Schalter gesucht wird (0-positive, 1-negative, 2-Achse wird nicht referenziert)
richtung_achse3	- Richtung in der für die dritte zu referenzierenden Achse nach dem Schalter gesucht wird (0-positive, 1-negative, 2-Achse wird nicht referenziert)
richtung_achse4	- Richtung in der für die vierte zu referenzierenden Achse nach dem Schalter gesucht wird (0-positive, 1-negative, 2-Achse wird nicht referenziert)
achse1	- Nummer der ersten zu referenzierenden Achse (0=X, 1=Y, 2=Z, 3=C)
achse2	- Nummer der zweiten zu referenzierenden Achse (0=X, 1=Y, 2=Z, 3=C)
achse3	- Nummer der dritten zu referenzierenden Achse (0=X, 1=Y, 2=Z, 3=C)
achse4	- Nummer der vierten zu referenzierenden Achse (0=X, 1=Y, 2=Z, 3=C)

LIBSPEC int __stdcall ParameterLesen(NCStepCommPortSettings *pNCStepCommPortSettings, unsigned char Hbyte, unsigned char Lbyte, unsigned char *daten);

Die Funktion „ParameterLesen“ liefert den angeforderten Parameter in dem übergeben Datenfeld zurück. Die Daten müssen vom Benutzer in den zum Parameter passenden Datentyp (z.B. long, float, char *) umgewandelt werden (casten).
Parameter:

// Hbyte	- aus den #define HBYTE_...
// Lbyte	- aus den #define LBYTE_...
// daten	- Pointer auf Array mit 16 Elementen von Typ „char“.

LIBSPEC int __stdcall ParameterAendern_ControllerName(NCStepCommPortSettings *pNCStepCommPortSet-tings, unsigned char Hbyte, unsigned char Lbyte,^char *cname);

Parameter:

Hbyte	- aus den #define HBYTE_...
Lbyte	- aus den #define LBYTE_...
cname	- Controllername, GENAU 16 char

Verwendung der Funktionen in Visual Basic

Visual Basic kennt keine Pointer, statt dessen werden die Variablen „by Reference“ übergeben. Die Funktionen werden nicht über ihren Namen, sondern über einen Alias angesproche. Dieser setzt sich zusammen aus einem Unterstrich, dem Funktionsnamem und einem @, hinter dem die Anzahl der übergebenen Bytes angehängt wird.

Beispiele:

```
Option Explicit
Public Declare Function NCStepOpenConnectionPort Lib „C:\WINDOWS\system32\SendeFrameDLL.dll“ _
Alias „_NCStepOpenConnectionPort@8“ (ByRef objTest As test, ByVal number As Integer) As Integer

Public Declare Function NCStepCloseConnection Lib „C:\WINDOWS\system32\SendeFrameDLL.dll“ _
Alias „_NCStepCloseConnection@4“ (ByRef objTest As test) As Integer

Public Declare Function BetriebsArtenWechsel Lib „C:\WINDOWS\system32\SendeFrameDLL.dll“ _
Alias „_BetriebsArtenWechsel@16“ (ByRef objTest As test, ByVal ziel_ba As Byte) As Integer

Public Declare Function FahreRelativ_XY_mm Lib „C:\WINDOWS\system32\SendeFrameDLL.dll“ _
Alias „_FahreRelativ_XY_mm@32“ (ByRef objTest As test, ByVal BahnVf As Single, _
ByVal AnzKoordPaare As Byte, ByRef xf As koordarr, ByRef yf As koordarr, _
ByVal rampenbits As Byte) As Integer
```