



Manual de usuario para el uso de la librería dll para Microsoft C++ y Borland Builder C++.

Man Micropap.dll
Rev. 1.4
14/11/2005
Autor: Ferran Sanabria Ortega
www.microPaP.com

Esta documentación explica como utilizar la libreria de vinculos dinámicos “micropap.dll” a través de Microsoft VisualC++™, y Borland C++ Builder.

1. Referencia a las funciones que puedes utilizar

char IniComunicationsPaP

void EnviarTrama

char TramaEnviada

char TramaRecibida

void EndComunicationsPaP

2. Ejemplo

Inicializa las comunicaciones RS-232

Declaración

```
char IniComunicacionesPaP (char *port_arg, int rate_arg, serial_parity parity_arg, char  
ByteSizechar)
```

Parametros

*char *port_arg*

Puntero a char. Especifica el nombre del puerto de comunicaciones a utilizar.

int rate_arg

Entero. Especifica la velocidad de comunicaciones.

serial_parity parity_arg

```
enum serial_parity { spNONE, spODD, spEVEN };
```

Paridad. Especifica la paridad a utilizar en las comunicaciones

char ByteSizechar

Caracter. Especifica el tamaño del byte de datos.

Uso

```
char error;
```

```
error = IniComunicacionesPaP ("COM1", 19200, spNONE, 8);
```

Retorna

0 Ningún error

-1 Error al conectar el dispositivo de comunicaciones

-2 No se ha podido crear el dispositivo de comunicaciones

Explicación

Esta función inicializa el puerto de comunicaciones y lo deja preparado para su uso. Se debe llamar a esta función para poder comunicar a través del puerto RS-232.

EnviarTrama

Envía una trama o comando por el puerto de comunicaciones

Declaración

```
void EnviarTrama(char identificador, int N_Pasos_Totals, char Sentit_Gir, char N_Pasos_Aceleracio, char N_Pasos_Deceleracio, char Velocitat_Max, char Velocitat_Min, char MicroPasos, char CorrentStop, char SearchHome, char ContinuousMov, char Prescaler);
```

Parametros

char identificador

Caracter. Especifica el identificador del motor. El primer byte de la trama a enviar a la controladora es el que identifica el motor al que va dirigida la trama. En el caso de la controladora *MPAP-1AXVI* al ser una conexión punto a punto este byte es fijo y el valor que debe tener es 0x30 . Este podrá tener diferentes valores si se trabaja a través de la controladora de 10 motores MPAP-HOST-6AXV1 . (0x30 .. 0x39)

int N_Pasos_Totals

Entero. Especifica el número de pasos a efectuar en regimen estable.

char Sentit_Gir

Caracter. Especifica el sentido de giro. Se deberá enviar el código 0x82 para girar en sentido de las agujas del reloj (CW, clock wise) o 0x76 para girar en sentido contrario al de las agujas del reloj (CCW, counter clock wise).

char N_Pasos_Aceleracio

Caracter. Especifica el número de pasos que realizará el motor en la rampa de aceleración para alcanzar la velocidad de regimen. El valor está limitado entre 0 y 99 (0x00+0x30 .. 0x63+0x30). Se debe sumar 0x30 al valor enviado.

char N_Pasos_Deceleracio

Caracter. Especifica el número de pasos que realizará el motor en la rampa de deceleración para alcanzar la velocidad mínima antes de la parada . El valor está limitado entre 0 y 99 (0x00+0x30 .. 0x63+0x30). Se debe sumar 0x30 al valor enviado.

char Velocitat_Max

Caracter. Especifica la velocidad máxima a la que se moverá el motor en funcionamiento estable o de regimen. El valor está limitado entre 0 y 200 (0x00+0x30 .. 0xC8+0x30). Se debe sumar 0x30 al valor enviado.

char Velocitat_Min

Caracter. Especifica la velocidad a la se moverá el motor antes de pararse. El valor está limitado entre 0 y 200 (0x00+0x30 .. 0xC8+0x30). Se debe sumar 0x30 al valor enviado.

char MicroPasos

Caracter. Especifica el número de micropasos que se pueden realizar entre pasos. Los valores que puede tomar son :

- 0x30 : pasos completos
- 0x31 : 1/2 pasos
- 0x32 : 8 micropasos entre pasos
- 0x33 : 16 micropasos entre pasos
- 0x34 : 32 micropasos entre pasos
- 0x35 : 64 micropasos entre pasos

char CorrentStop

Caracter. El byte 10 indica la corriente que se desea que se le suministre al motor cuando este parado. Esta corriente (indicada en mA) permitirá mantener el par al motor en las paradas. El valor está limitado entre 0 y 63 (0x00+0x30 .. 0x3F+0x30). Se debe sumar 0x30 al valor enviado.

char SearchHome

Caracter. Búsqueda de los finales de carrera FC1 o FC2

- 0x30 = Testea FC durante los movimientos.
- 0x31 = Testea FC1 girando derecha
- 0x32 = Testea FC2 girando izquierda
- 0x33 = No testea los FC durante los movimientos.

char ContinuousMov (modo infinito)

Caracter. Especifica si se quiere realizar un movimiento continuo sin atender al número de pasos la trama. Los valores que puede tomar son :

0x30 : Detiene el movimiento en continuo. Unicamente tiene sentido si se está ejecutando un movimiento en continuo. Para el movimiento en continuo.

0x31 : Inicia el movimiento en continuo.

char Prescaler

Caracter. Especifica el prescaler de velocidad. Los valores que puede tomar son :

- 0x30 = 1/2 (máxima velocidad)
- 0x31 = 1/4

- 0x32 = 1/8
- 0x33 = 1/16
- 0x34 = 1/32
- 0x35 = 1/64
- 0x36 = 1/128 (valor por defecto)
- 0x37 = 1/256 (mínima velocidad)

Uso

EnviarTrama (0x32, 400, 0x52, 0x30, 0x30, (char) 0xfa+0x30, 0x30, 0x33, 0x30, 0x33, 0x30, 0x35);

Retorna

No retorna ningún valor. Para saber el resultado del envío se debe consultar el buffer de transmisión y recepción.

Explicación

Esta rutina envía una trama a través del puerto de comunicaciones para ejecutar un determinado comando.

TramaEnviada

Consulta la información enviada por el puerto de comunicaciones.

Declaración

```
char TramaEnviada (char *BufferTxUser)
```

Parametros

*char *BufferTxUser*

Puntero a char. Puntero a caracter a partir del cual se retornarán los bytes que hayan sido enviados por el puerto de comunicaciones.

Uso

```
#define MAX_CAR 128
```

```
unsigned char BufferTx [MAX_CAR];
```

```
char IndTx;
```

```
IndTx = TramaEnviada((char *) &BufferTx[0]);
```

Retorna

Retorna el número de bytes que han sido enviados por el puerto de comunicaciones

Explicación

Esta función consulta la información enviada por el puerto de comunicaciones y retorna los bytes enviados a partir del puntero a char que se le pasa como parámetro.

TramaRecibida

Consulta la información recibida por el puerto de comunicaciones.

Declaración

```
char TramaRecibida (char *BufferRxUser)
```

Parametros

*char *BufferRxUser*

Puntero a char. Puntero a caracter a partir del cual se retornarán los bytes que hayan sido recibidos por el puerto de comunicaciones.

Uso

```
#define MAX_CAR 128
```

```
unsigned char BufferRx [MAX_CAR];
```

```
char IndRx;
```

```
IndRx = TramaRecibida((char *) &BufferRx[0]);
```

Retorna

Retorna el número de bytes que han sido recibidos por el puerto de comunicaciones

Explicación

Esta función consulta la información recibida por el puerto de comunicaciones y retorna los bytes recibidos a partir del puntero a char que se le pasa como parámetro.

EndComunicationsPaP

Finaliza las comunicaciones RS-232

Declaración

```
void EndComunicationsPaP (void);
```

Parametros

Uso

```
EndComunicationsPaP ();
```

Retorna

Explicación

Esta función cierra el puerto de comunicaciones y libera la memoria asociada. Se debe llamar a esta función para liberar las comunicaciones del puerto RS-232.

Ejemplo

```
#include "stdafx.h"
//-----
//#pragma hdrstop
#include <conio.h>
#include <stdio.h>
//-----
#define MAX_CAR 128

enum serial_parity { spNONE, spODD, spEVEN };

extern char IniComunicacionesPaP (char *port_arg, int rate_arg, serial_parity parity_arg,
char ByteSizechar);

extern void EnviarTrama(char identificador, int N_Pasos_Totals, char Sentit_Gir, char
N_Pasos_Aceleracio,char N_Pasos_Deceleracio, char Velocitat_Max, char
Velocitat_Min, char MicroPasos, char CorrentStop, char SearchHome,char
ContinuousMov, char Prescaler);

extern void EndComunicacionesPaP (void);

extern char TramaRecibida (char *BufferRxUser);

extern char TramaEnviada (char *BufferTxUser);

unsigned char BufferRx [MAX_CAR]; // buffer de recepción

unsigned char BufferTx [MAX_CAR]; // buffer de transmisión

int IndRx , IndTx ; //índices recepción y transmisión

int main(int argc, char* argv[])
{
int i,error;

/* inicializa comunicaciones :

puerto : COM1
velocidad : 19200 bps
paridad : sin paridad, spNONE
bits de datos : 8
*/

error=IniComunicacionesPaP ("COM1", 19200, spNONE, 8);

/* Envía una trama :
```

```

identificador = 0x32;
N_Pasos_Totals = 400 pasos;
Sentit_Gir = 0x52;
N_Pasos_Aceleracio = 0x00;
N_Pasos_Deceleracio = 0x00;
Velocitat_Max = 0xfa;
Velocitat_Min = 0x00;
MicroPasos = 0x03 : 32 micropasos
CorrentStop = 0x00;
SearchHome = 0x03; No testea FC durante los movimientos
ContinuousMov = 0x00; Para el movimiento continuo
Prescaler = 0x05 ;
*/

EnviarTrama (0x32, 400, 0x52, 0x00+0x30, 0x00+0x30, (char) 0xfa+0x30,
0x00+0x30, 0x03+0x30, 0x0+0x30, 0x03+0x30, 0x00+0x30, 0x05+0x30);

printf ("una tecla para continuar\n");
getch();

IndRx = 0;
IndTx = 0;

/* Carga el buffer de transmisión interno a partir de &BufferTx[0] */

IndTx= TramaEnviada((char *) &BufferTx[0]);

for (i=0;i<IndTx;i++) printf ("E: %02x\n", BufferTx[i]);

printf ("-----\n");

/* Carga el buffer de recepcion interno a partir de &BufferRx[0] */

IndRx= TramaRecibida((char *) &BufferRx[0]);

for (i=0;i<IndRx;i++) printf ("R: %02x\n", BufferRx[i]);

/* Libera el canal de comunicación */

EndComunicationsPaP () ;

printf ("una tecla para continuar\n");
getch();

return 0;
}

```