NJSDK ™

µ §

Chinese Software Development Kits

Version 3.0

µ §µ §

1. Introduction

NJSDK is a C Library based on NJSTAR Version 3.0 Plus. It contains all the functions for handling Chinese input, display and printing. With NJSDK, user can develop any Chinese applications without knowing how Chinese characters are inputted, displayed or printed. It is as easy as writing an English program.

NJSDK provides a Chinese interface for any C program. It reserved the bottom line for Chinese input, and all other 24 lines (on VGA, 18 on EGA, 29 on SVGA) are working area for the application. NJSDK starts by loading Chinese font and dictionaries into memory, which require 200KB. If user select to leave font on disk, then NJSDK only requires 80KB to operate. When NJSDK exits, all used memory are released, and screen is back to text mode.

NJSDK supports all the input methods in NJSTAR v3.0, user defined methods also supported in the same way as in NJSTAR. ALT + Function Keys are reserved by NJSDK for switching between the Chinese input methods, and other Chinese related function.

Mouse operations are directly supported by NJSDK, application program can get mouse position by calling the NJSDK mouse functions.

Screen Saver and AutoSave facilities have been built in NJSDK. Active interval can be set by a configuration program NJSDKCFG.EXE.

2. License Information

With the full payment for NJSDK, the developers are granted a distribution license which permits the distribution of all NJSTAR dictionaries, drivers and fonts (total 30 files) with their application without any further royalty payment. **Distribution of any Executable file (*.EXE) of NJSTAR is strictly prohibited**. Each licensed copy has an unique serial number, this protects any illegal use of NJSDK by other un-licensed users. Licensed user are entitled a free upgrade to next version of NJSDK when it becomes available. Please contact the author for current price. Other profit sharing arrangements can be made at request. Please contact the author if you or your company have any special requirements and cooperations.

LIMITED WARRANTY

A) The Author of NJSDK warrants that all disks provided are free from defects in material and workmanship, assuming normal use, for a period of 30 days from the date of purchase.

B) The Author of NJSDK warrants that the program will perform in substantial compliance with the documentation supplied within this document. If a significant defect in the product is found, the Purchaser may return the product for a refund. In no event will such a refund exceed the purchase price of the product.

C) Use of this product for any period of time constitutes your acceptance of this agreement

and subjects you to its contents.

D) EXCEPT AS PROVIDED ABOVE, THE AUTHOR OF NJSDK DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE PRODUCT SHOULD THE PROGRAM PROVE DEFECTIVE, THE PURCHASER ASSUMES THE RISK OF PAYING THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL THE AUTHOR OF NJSDK BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING WITHOUT LIMITATION DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OR THE INABILITY TO USE THIS PRODUCT EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

3. List of files in NJSDK package

NJSDK.LIB	The NJSDK C subroutine library (MSC 7.0 Large model)
NJSDK.H	The header file for NJSDK
KEYS.H	The header file for key and mouse button definitions
******.C	Example programs
CC.BAT	Batch file used to compile a program, eg. CC hello
NJSDK.DOC	NJSDK user manual in MS-Word 2.0 for Windows Format
NJSDK.TXT	Text version of NJSDK.DOC
NJSDK.CFG	NJSDK configuration data file
NJSDK.INI	Define the location of NJTSTAR's dictionaries and font
NJSDCFG.EXE	The NJSDK configuration program
SUPER.VGA	The super VGA definition file (required by NJSDKCFG)

Following files from NJSTAR are needed for program based on NJSDK to run. Those files can be distributed with user's .EXE file **only after a distribution license has been purchased**.

CCLIB.16	16x16 Simplified Chinese bitmap
CCLIB.24	24x24 Simplified Chinese bitmap (With registered version)
CCLIBF.16	16x16 un-Simplified Chinese bitmap
CCLIBF.24	24x24 un-Simplified Chinese bitmap (With Registered version)
ASCII.16	16x8 Ascii bitmap
ASCII.24	24x12 Ascii bitmap
NANJIPY.DIC	NJSTAR PinYin dictionary
NANJILX.DIC	NJSTAR Lianxiang dictionary
NJINPUT*.DIC	External character input methods dictionary
NJCIZU*.DIC	External word input methods dictionary
TOLERATE.DIC	PinYin Tolerance lists (text file)
********.DRV	Printer drivers

If NJSTAR is installed on C:\NJSTAR, a file NJSDK.INI can be created in the same directory as user's application. It should contains two lines:

NJDIC=C:\NJSTAR
NJZK=C:\NJSTAR

NJSDK will try to locate the font and dictionaries in the same directory as users EXE file. If not found, NJSDK will try those as defined in NJSDK.INI. user can also set two DOS environment variables in AUTOEXEC.BAT to specify the location of fonts and dictionaries (override NJSDK.INI).

SET NJZK=C:\NJSTAR
SET NJDIC=C:\NJSTAR

In this way, only one set of font and dictionaries are needed on one computer, all the NJSDK applications and NJSTAR can access the same font and dictionaries.

4. Function Keys Reserved by NJSDK

Alt+F1 - NJSDK Help	Alt+F9 - Pure Chinese Input
Alt+F2 - Other Input Methods	Alt+F10 - FanTiZi or JianTiZi
Alt+F3 - QuWeiMa Input	Alt+F11 - Adding Character/Word
Alt+F4 - GuoBiao Input	Alt+F12 - Remove Character/Word
Alt+F5 - PinYin Input	
Alt+F6 - ASCII Input	Alt+'X' - LianXiang
Alt+F7 - ZhuYin Input	Alt+'Z' - Get PinYin/ Input Code
Alt+F8 - Options	ESC - Cancel Current Input

5. List of NJSDK Functions

Function Name	Descriptions
NJ_AutoSave	Tell NJSDK which is your AutoSave function
NJ_box	Draw a rectangle box in given color
NJ_button	Select from a list of buttons
NJ_clscolum	clear a column of text
NJ_clsrows	clear rows of text
NJ_cursorOnOff	turn on / off the blinking input cursor
NJ_dialog	A dialog entry program
NJ_end	end of NJSDK session (back to text mode, unload font and dictionaries)
NJ_entry	entry input routine
NJ_fillwin	fill a window with color
NJ_formfeed	Form Feed
NJ_fontsize	Set the size of Chinese Characters
NJ_getkey	get the key pressed by user
NJ_getmouse	get mouse position
NJ_gets	get a string with Chinese characters
NJ_gets0	get a string without Chinese characters
NJ_getvideo	get NJSDK video screen configuration
NJ_init	Initialise NJSDK library (setup Graphics mode, load Font and Dictionaries)

NJ_line	Draw a horizontal or vertical line
NJ_linefeed	Line feed
NJ_menu	menu selection routine
NJ_mousecursor	turn on /off the mouse cursor
NJ_prt_end	end of the printing
NJ_prt_init	Initialising printer
NJ_prts	print a string to printer
NJ_putc	put a English (ascii 1-255) to screen
NJ_puts	put a string on screen (with or without Chinese Characters)
NJ_setcursor	set cursor position
NJ_setinput	set Chinese input Method
NJ_setmargin	Set Margins and line spacing
NJ_window	draw a window with color

6. Detailed Description of NJSDK functions

- ## Initialization

NJ_init Initialise NJSDK library, such as setup graphics mode, load Chinese font and dictionaries. This function must be called before any NJSDK function is used.

Calling Syntax: NJ_init(char *homedir)

Input: homedir - a char pointer points to the dir where application is started
this is used for NJSDK to locate font and dictionaries

Return: 0 - Initialisation OK
-1 - Initialisation failed

See also: NJ_end

Example:

```
main(int argc, char **argv)
{
        NJ_init(argv[0]);
        ....
        ....
        NJ_end();
}
```

NJ_end Exit NJSDK library, font and dictionaries are unloaded and screen are back to text mode.

Calling Syntax: NJ_end()

Input: (none)

Output: (none)

Return: (none)

See also: NJ_init

Example: see NJ_init

NJ_AutoSave Tell NJSDK which is your AutoSave function

Calling Syntax: int NJ_AutoSave(int(*asfun)())
Input: int *asfun() -- a function pointer points to the user's SaveSave Function.
Output: (none)
Example:

```
int myautosave()
{
   /* Close the data base, or    */
   /* do anything you like to do */
}

main(int argc, char **argv)
{
   NJ_init(argv[0]);
   NJ_AutoSave(myautosave);
   ....
   ....
   NJ_end();
}
```

NOTE: User should call this function if Autosave features are needed, and do any thing you want to do in it, such as backup data base. NJSDK will call this function in a preset interval or when any emergency happen. a default function is provided by NJSDK, which does nothing.

NJ_getvideo Get Video screen parameters

Calling Syntax: int NJ_getvideo(struct NJVIDEO *njv)

Input: struct NJVIDEO *njv -- pointer to NJVIDEO structure

```
struct NJVIDEO {
    int Videomode;     /* Current Video Mode (see NJSDK.H)    */
    int Nrows;         /* number of rows EGA=18 VGA=24 SVGA=29*/
    int BytesPerLine;  /* bytes per line EGA/VGA=80 SVGA=100  */
    int PixelsPerRow;  /* Pixels per row EGA=18 VGA/SVGA=19   */
    int Mono;          /* Is a monochrome mornitor            */
};
```

Output: struct NJVIDEO *njv

See also: NJ_init()
Example:

```
#include "njsdk.h"
struct NJVIDEO njv;
main(int argc, char **argv)
{
      NJ_init(argv[0]);
      NJ_getvideo(&njv)
      ....
      ....
```

```
        NJ_end();
    }
```

- ## Keyboard and Mouse Input

NJ_getkey wait for user to press a key, and return the key values as defined in header file KEYS.H

Calling Syntax: NJ_getkey()
Input: (none)
Output: (none)
Return: the key pressed (see KEYS.H for predefined value)
See also: NJ_getmouse()
Example:

```
#include "keys.h"
....
k=NJ_getkey();
if(k==ESC) exit(0);
....
```

NJ_getmouse report the current mouse position (row, column in pixels).

Calling Syntax: NJ_getmouse(int *row, int *col)
Input: (none)
Output: int *row, *col -- the current mouse cursor position
Return: the key value if any key is pressed
See also: NJ_getkey()
Example:

```
int row, col;
NJ_getmouse(&row, &col);
sprintf(str,"mouse position: %d %d", row,col);
NJ_puts(str, ...);
```

NJ_mousecursor Turn on / off mouse cursor during screen outputs

Calling Syntax: NJ_mousecursor(int on)
Input: on = 1 -- turn ON the mouse cursor
on = 0 -- turn OFF the mouse cursor
Output: (none)
Return: (none)
See also: NJ_getmouse()
Example:

```
NJ_mousecursor(0);
NJ_puts(str, ...);
NJ_mousecursor(1);
....
```

NJ_cursorOnOff turn ON / OFF the blinking input cursor

Calling Syntax: NJ_cursorOnOff(int on)
Input: on = 1 -- turn ON the cursor

on = 0 -- turn OFF the cursor
Output: (none)
Return: (none)
See also: NJ_getkey()
Example:

```
NJ_cursorOnOff(1);
NJ_gets(str, ...);
NJ_cursorOnOff(0);
....
```

NJ_setcursor set the blinking cursor position

Calling Syntax: NJ_setcursor(int row, int col)
Input: row, col -- the new cursor position in character coordinate (25x80)
Output: (none)
Return: (none)
See also: NJ_cursorOnOff()
Example:

```
NJ_setcursor(24, 1);
```

NJ_gets get a string including Chinese characters from screen

Calling Syntax: int NJ_gets(char *str, int row, int col, int len, int Fcolor, int Bcolor)
Input: str -- the buffer for holding the input string, minimum length = len.
len -- the maximum len of input string
row, col -- the screen position where input string starts
Fcolor -- foreground color (0 - 15)
Bcolor -- background color (0 - 15)
Output: str -- the string entered by user
Return: -1 -- user cancel the input by ESC
0 -- user end the input by ENTER
>0 -- other non-editing keys
See also: NJ_puts()
Example:

```
char str[80];
NJ_gets(str, 10,5, 70, 15, 1);
```

NJ_gets0 Same as NJ_gets, except that Chinese Input are not allowed in this function

---

NJ_setinput set Chinese input method

Calling Syntax: int NJ_setinput(int method, char select)
Input: method = 0 -- ascii input
1 -- PinYin input
2 -- QuWeiMa input
3 -- GuoBiao Input
4 -- Other inputs
5 -- ZhuYin input
select -- for select a sub-method if the method has multiple choose such as 4 and 5. Not used for methods 0-3.
Output: (none)

Return: (none)
See also:
Example:

```
/* select WuBiZiXin method */
NJ_setinput(4, 'B');
```

- **Screen Display**

NJ_puts Display a string of ascii, Chinese or mix of them on specific position of the screen with specific color.
Calling Syntax: int NJ_puts(char str, int row, int col, int Fcolor, int Bcolor)
Input: str -- the string to display (ascii, Chinese or mix of them)
row, col -- the screen position where the output starts
Fcolor -- foreground color (0 - 15)
Bcolor -- background color (0 - 15)
Output: (none)
Return: (none)
See also: NJ_gets()
Example:

```
/* display 'Chinese char' on 20,10 with Yellow on Red*/
NJ_puts("µ §", 20,10, 14,4);
```

NJ_putc Display a ascii character on specific position of the screen with color.
Calling Syntax: int NJ_putc(char c, int row, int col, int Fcolor, int Bcolor)
Input: c -- the character to display (ascii 1-255)
row, col -- the screen position where the output starts
Fcolor -- foreground color (0 - 15)
Bcolor -- background color (0 - 15)
Output: (none)
Return: (none)
See also: NJ_puts()
Example:

```
/* display 'A' on 20,10 with Yellow on Red*/
NJ_putc('A', 20,10, 14,4);
```

NJ_window Draw a outline window with specific color

Calling Syntax: int NJ_window(int r1, int c1, int h, int v, int color)
Input: r1, c1 -- upper left corner of the window (in pixels)
h, v -- the Horizontal and vertical size of the window (in pixels)
color -- the color of the screen
Output: (none)
Return: (none)
See also: NJ_fillwin()
Example:

```
NJ_window(50,100, 100, 400, 2);
```

NJ_fillwin Same as NJ_window, except not boarder outline

Calling Syntax: int NJ_fillwin(int r1, int c1, int h, int v, int color)
Input: r1, c1 -- upper left corner of the window (in pixels)
h, v -- the Horizontal and vertical size of the window (in pixels)
color -- the color of the screen
Output: (none)
Return: (none)
See also: NJ_window()
Example:

```
NJ_fillwin(50,100, 100, 400, 2);
....
```

NJ_clscolum Clear portion of the line with specified color

Calling Syntax: int NJ_clscolum(int row1, int col1, int col2, int color)
Input: row -- the line number
col1 -- the start column number
col2 -- the end column number
color -- the color
Output: (none)
Return: (none)
See also: NJ_clsrows()
Example:

```
NJ_clscolum(1, 0,80, 15);
....
```

NJ_clsrows Clear number of lines

Calling Syntax: int NJ_clsrows(int row1, int row2, int color)
Input: row1 -- the start line number
row2 -- the end line number
color -- the color
Output: (none)
Return: (none)
See also: NJ_clscolum()
Example:

```
NJ_clsrows(5,10, 15);
....
```

NJ_line Draw a horizontal or vertical line

Calling Syntax: int NJ_line(int row1, int col1, int row2, int col2, int color)
Input: row1 -- the start row in pixel
col1 -- the start column in pixel
row2 -- the end row in pixel
col2 -- the end column in pixel
color -- the line color
Output: (none)

Return: (none)
See also: NJ_box()
Example:

```
NJ_line(5,0,5,640,15);
....
```

NJ_box Draw a rectangle box in given color

Calling Syntax: int NJ_line(int row1, int col1, int row2, int col2, int color)
Input: row1 -- the start row in pixel
col1 -- the start column in pixel
row2 -- the end row in pixel
col2 -- the end column in pixel
color -- the line color
Output: (none)
Return: (none)
See also: NJ_line()
Example:

```
NJ_box(5,0,50,640,15);
....
```

NJ_menu Selecting a list of options by menu

Calling Syntax: int NJ_menu(int Fcolor1, int Bcolor1, int Fcolor2,int Bcolor2,int pressed,
struct MENU_S *sl,int ns,int ini,int(*msgfun)(char *))

Input: Fcolor1, Bcolor1 -- the foreground and background color of normal text
Fcolor2, Bcolor2 -- the foreground and background color of high lighted text (0 - 15).
pressed=0 -- mouse cursor will move the selection item without press the mouse button
pressed=1 -- mouse cursor will move the selection item only if mouse buttons are pressed
ns -- number of selections in sl
ini -- the initial selection
msgfun() -- the pointer to a user provided function for display the message
sl -- pointer to the structure of menu items.

```
struct MENU_S { /* as defined in NJSDK.H */
    char row;       /* display row */
    char col;       /* display col */
    char len;       /* display background len */
    char list;      /* the selection key in upper case */
    char *str;      /* menu string */
    char *msg;              /* message associated */
} ;
```

Output: (none)
Return: the number of selected item

See also: NJ_entry()
Example:

```
putmsg(char *str)
{
        NJ_puts(str, 20,20, 2,15);
}
struct MENU_S sl[2]={
        {20,10, 10, 'Y', "Yes", "Confirm"},
        {21,10, 10, 'N', "No ", "quit"}
}
main()
{
   ....
        NJ_menu(11, 2, 14,4, 0, sl,2,0, putmsg);
   ....
}
```

NJ_button Select a list of options by pressing the button

Calling Syntax: int NJ_button(struct MENU_S *sl, int ns, int ini, int pressed)

Input: pressed=0 -- mouse cursor will move the selection item without press the mouse button
pressed=1 -- mouse cursor will move the selection item only if mouse buttons are pressed
ns -- number of selections in sl
ini -- the initial selection
sl -- pointer to the structure of menu items.

```
struct MENU_S { /* as defined in NJSDK.H */
    char row;       /* display row */
    char col;       /* display col */
    char len;       /* display background len */
    char list;      /* the selection key in upper case */
    char *str;      /* menu string */
    char *msg;      /* message associated, not used by this function*/
} ;
```

Output: (none)
Return: the number of selected item
See also: NJ_menu()
Example:

```
struct MENU_S sl[2]={
        {20,10, 10, 'Y', "Yes", " "},
        {21,10, 10, 'N', "No ", " "}
}
main()
{
   ....
        NJ_button(sl,2,0,1);
   ....
}
```

NJ_entry data entry routine

Calling Syntax: int NJ_entry(int Fcolor1, int Bcolor1, int Fcolor2,int Bcolor2,
struct ENTRY_S *sl,int ns,int ini,int(*msgfun)(char *))

Input: Fcolor1, Bcolor1 -- the foreground and background color of normal text
Fcolor2, Bcolor2 -- the foreground and background color of high lighted text (0 - 15).
ns -- number of selections in sl
ini -- the initial selection
sl -- pointer to the structure of entry items.

```
struct ENTRY_S {
    char row;          /* display row */
    char col;          /* display row */
    char len;          /* len of the  field */
    char type;         /* type of the field, not used yet*/
    char str[80];      /* str to hold field */
    char *msg;         /*message associated*/
};
```

msgfun(char *msg) -- the pointer to a user provided function for display the message.
INPUT: msg -- the string of message

Output: sl -- the data entered are stored in sl[i].str.
Return: 0 -- Normal return
-1 - Data entry is cancelled by user
See also: NJ_menu
Example:

```
putmsg(char *str)
{
    NJ_puts(str, 20,20, 2,15);
}
struct ENTRY_S sl[2]={
    20,10, 10,0, " ", "Enter Your Name",
    21,10, 10,0, " ", "Enter you age"
}
main()
{
  ....
    NJ_entry(11, 2, 14,4, sl,2,0, putmsg);
  ....
}
```

NJ_dialog data entry with push button and radio buttons

Calling Syntax: int NJ_dialog(int Fcol1,int Bcol1,int Fcol2,int Bcol2,int Fcol3,
int Bcol3, struct DIALOG_S *sl,int ns,int ini,
int(*DialogProc)(int, int), int(*msgfun)(char *));

Input: Fcol1, Bcol1 -- the foreground and background color of normal text
Fcol2, Bcol2 -- the foreground and background color of data entry non-current item (0 - 15)

Fcol3, Bcol3 -- the foreground and background color of high lighted data entry item (0 - 15)

ns -- number of selections in sl
ini -- the initial selection
sl -- pointer to the structure of dialog items.

```
struct DIALOG_S {
   char row;          /* display row */
   char col;          /* display row */
   char len;          /* len of the  field */
   char type;         /* type of the field:  0  - data entry */
                      /*                    -1 - push button */
                      /*                    -2 - radio button */
   char radio;        /* store radio status*/
   char str[80];      /* str to hold field value*/
   char *msg;         /*message associated*/
};
```

msgfun(char *msg) -- the pointer to a user provided function for display the message.
INPUT: msg -- the string of message
DialogProc(int k, int s) -- the pointer to a user provided function for handling the dialog messages.
INPUT: k=-1 -- initialize the dialog parameters
k=0 -- button number S has been pressed,
k>0 -- a non-editing key has been pressed
Output: -1 -- Cancel the dialog box
0 -- End dialog and return to calling program
1 -- Re-paint all entries

NOTE: In DialogProc, user can access the information in DIALOG_S structure -- sl. See *****.C for example usage.

Output: sl -- the data entered are stored in sl[i].str.
Return: 0 -- Normal return, entry valid
-1 - Data entry is cancelled by ESC or a push button.
See also: NJ_menu
Example:

```
#include <stdio.h>
#include "njsdk.h'
#include "keys.h'

struct ENTRY_S sl[4]={
      {10,10, 10,0,0 " ", "Enter Your Name"},
      {12,10, 10,0,0 " ", "Enter your age"},
      {15,10, 10,-1,0 "OK",     "Accept the input"},
      {15,13, 10,-1,0 "Cancel", "Quit the dialog"}
}
int putmsg(char *str)
{
      NJ_puts(str, 20,20, 2,15);
}
int mydialogproc(int k, int s)
```

```
{
        if(k<0){
                if(s==2) return (0);      /* Ok     */
                else if(s==3) return(-1); /* cancel */
        }else if(k==PGUP){GetLastRecordFromDataBase();}
        }else if(k==PGDN){GetNextRecordFromDataBase();}
        return(1);
}
main()
{       ....
        i=NJ_entry(15,1,11,2,14,4,sl,2,0,mydialogproc, putmsg);
        if(i<0)         /* Cancelled */
        else if(i>=0)   /* Ok       */
}
```

• Printer Output (Not included in trial version)

NJ_prt_init Initialise printer driver

Calling Syntax: NJ_prt_init()
Input: (none)
Return: 0 - Initialisation OK
-1 - Initialisation failed
See also: NJ_init()
Example:

```
main(int argc, char **argv)
{
        NJ_init(argv[0]);
        ....
        NJ_prt_init()
        NJ_prts("CHINESE string");
        ....
        NJ_end();
}
```

NJ_prt_end end of the printing

Calling Syntax: NJ_prt_end()
Input: (none)
Return: 0 - Ok
-1 - failed
See also: NJ_prt_init()
Example:

```
main(int argc, char **argv)
{
NJ_init(argv[0]);
....
NJ_prt_init()
NJ_prts("CHINESE string");
NJ_prt_end()
....
NJ_end();
}
```

NJ_setmargin Set Margins and line spacing

Calling Syntax: int NJ_fontsize(int size)
Input: Size - size of printed Chinese chars (=1 or 2)
Output: (none)
Return: (none)
See also: NJ_prts()
Example:

```
NJ_fontsize(2);
.....
```

NJ_setmargin Set Margins and line spacing

Calling Syntax: NJ_setmargin(int l, int r, int t, int b, int s)
Input: l = left margin in number of Chinese characters
r = right margin in number of Chinese characters
t = top margin in number of lines (1 line = 30 dots)
b = bottom margin in number of lines (1 line = 30 dots)
s = line spacing in number of pixels/dots
Return: 0 - Ok; -1 -Failed.
See also: NJ_linefeed()
Example:

```
NJ_setmargin(5,5,2,6,12);
....
```

NJ_prts Display a string of ascii, Chinese or mix of them on specific position of the screen with specific color.

Calling Syntax: int NJ_prts(char str)
Input: str -- the string to display (ascii, Chinese or mix of them)
Output: (none)
Return: (none)
See also: NJ_puts()
Example:

```
NJ_prts("CHINESE Chars");
```

NJ_linefeed feed the printer by number of lines

Calling Syntax: NJ_linefeed(int pixels)
Input: pixels -- the number of pixels to be feed
Return: (none)
See also: NJ_formfeed()
Example:

```
NJ_linefeed(5);
```

NJ_formfeed feed the printer paper to next page

Calling Syntax: NJ_formfeed()
Input: (none)
Return: (none)
See also: NJ_linefeed()
Example:

```
NJ_formfeed();
....
```
