

NBASMC: NewBasic Library
NewBasic Assembler
Version 00.91.04
A Beta version
Freeware

Forever Young Software
(C)opyright 1984-2001
All rights reserved
13 Feb 2001

Please note: Being that this is a beta version, some of this documentation may be incorrect. If there is a B (or *Beta*) in the first position of the line, that means that this sentence/paragraph does not pertain to this beta version, but to the final release when it is done.

Since this is a beta release, please use caution when using this library. Please let me know if you find any bugs or errors.

Get the latest version of this library at
<http://www.cybertrails.com/~fys/newbasic.htm>

Table of Contents

What is this library for	?
Description of the routines	?
Miscellaneous	?
Disclaimer	?
Bug report	?
Latest Version	?

What is this library for:

- This library is intended for the use with NBASM and the tiny model. When you used `.external` in your code, NBASM loads and searches for the specified symbol (procedure) within this library.

- See NBASM.DOC for more on how to use this library.

Description of the routines

- this is a list and description of the routines that are currently included in the NBASMC.LIB library file.
- This library does not contain some of the functions that you would think a library would need. This is because the code to create these functions would be quite a bit more than simply including them in your code as "straight" assembly. However, after time, I might include them just to make your programming a little easier.
- It is no where complete, so please be patient with me.

```
ARGCV --  command line interface (CPU needed: 8086)
subfunction
  = 00 = set delimiter char (default is '/')
        on entry: delimiter char
        on exit: nothing

        .186
        .external argcv
        push '-'
        push 0 ; function 0
        call argcv

  = 01 = get parameter count (should be called before function 2)
        on entry: segment of PSP
        on exit: ax = count

        .186
        .external argcv
        push cs
        push 1 ; function 1
        call argcv
        ; ax = number of parameters

  = 02 = get a parameter (function 1 above should be called first)
        on entry: number of parameter wanted
        on exit: if no carry: es:di -> parameter (asciiz)
                 if carry: error (count to high, or no param,
                 or function 1 not called yet)

        .186
        .external argcv
        push 3 ; get 3rd parameter
        push 2 ; function 2
        call argcv
        jc short error
        ; es:di -> parameter (asciiz)
```

```
BEEP -- Prints the BELL char (CPU needed: 8086)
        on entry: nothing
        on exit: nothing
```

```

.external beep
    call beep

*****
BLINK -- Enables/Disables the blinking attribute (CPU needed: 8086)
    on entry: (true|false)
    on exit:  nothing

.external blink
    push true                ; true > 0, false = 0
    call blink

*****
CAPSOFF -- Turns the caps lock off (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external capsoff
    call capsoff

*****
CAPSON -- Turns the caps lock on (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external capson
    call capson

*****
CURSOFF -- Turns the blinking cursor off (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external cursoff
    call cursoff

*****
CURSON -- Turns the blinking cursor on (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external curson
    call curson

*****
DELAY -- Delay for xx milliseconds (CPU needed: 8086)
    on entry: milliseconds to delay
    on exit:  nothing

.186
.external delay
    push 100h                ; delay for 100h milliseconds
    call delay

*****
FDFREE -- Get disk free space (CPU needed: 8086)
    on entry: drive

```

```

    on exit:  dx:ax = bytes free
              carry = status

.8086
.external fdfree
    push dx                ; dx = drive (0=current, 1=a, etc.)
    call fdfree

*****
FDTADATA -- Get DTA data of a filename given (CPU needed: 186)
    on entry: offset to asciiz filename
    on exit:  dx:ax = filesize
              bx = attrib
              cx = date
              si = time
              carry = status

.186
.external fdtadata
    push offset our_file  ; ( ds = segment of our_file )
    call fdtadata

*****
FREWIND -- move file pointer to first of file (CPU needed: 8086)
    on entry: handle
    on exit:  carry = status

.8086
.external frewind
    push bx                ; bx = opened handle
    call frewind

*****
GETCH -- Get a char from stdin without echo (CPU needed: 8086)
    on entry: nothing
    on exit:  al = char

.external getch
    call getch
    ; al = char

*****
GETCHE -- Get a char from stdin with echo (CPU needed: 8086)
    on entry: nothing
    on exit:  al = char

.external getche
    call getche
    ; al = char

*****
GETENV -- get environment value for variable passed (CPU needed: 8086)
    on entry: PSP segment, segment of variable, offset of variable
    on exit:  es:di -> string in environment

.186
.external getenv                ; search string is case sensitive

```

```
    push offset env_var    ; env_var db 'DIRCMD',0
    push ds                ; segment of above var
    push cs                ; segment of PSP
    call getenv            ;
    ; es:di -> string (if null, no match found)
```

```
GETSINTE -- Gets Signed integer from stdin (CPU needed: 8086)
  on entry: nothing
  on exit:  ax = signed integer
```

```
.external getsinte
  call getsinte
  ; ax = signed integer
```

```
GETVSEG -- returns the text video segment (CPU needed: 8086)
  on entry: nothing
  on exit:  ax = B000h or B800h
```

```
.external getvseg
  call getvseg
  ; ax = B000h or B800h
```

```
GRAPH -- includes the following sub functions (CPU needed: 8086)
  subfunction
```

```
    = 00 = get screen char attrib
      on entry: nothing
      on exit:  al = attrib
```

```
    .186
    .external graph
      push 00h
      call graph
      ; al = attrib
```

```
    = 01 = set screen char attrib
      on entry: attrib (word)
      on exit: nothing
```

```
    .186
    .external graph
      push 07h
      push 01h
      call graph
```

```
    = 02 = cls (clears entire screen)
      on entry: nothing
      on exit: nothing
```

```
    .186
    .external graph
      push 02h
      call graph
```

```
    = 03 = char out with attrib
```

```

    on entry: [sp+04] = char (word)
    on exit: nothing

.186
.external graph
    push 'A'
    push 03h
    call graph

= 04 = string out with attrib
    on entry: [sp+04] = offset to string (word)
    on entry: [sp+06] = segment to string (word)
    on exit: nothing

.186
.external graph
    push ds
    push offset temp_str
    push 04h
    call graph

*****
ISALPHA -- returns true|false if AL is alpha char (CPU needed: 8086)
    on entry: al = char
    on exit: zero flag = (true|false)

.external isalpha
    push ax
    call isalpha

*****
ISALT -- returns true|false if Alt key is depressed (CPU needed: 8086)
    on entry: nothing
    on exit: zero flag = (true|false)

.external isalt
    call isalt

*****
ISCNTRL -- returns true|false if AL is < 32 char (CPU needed: 8086)
    on entry: al = char
    on exit: zero flag = (true|false)

.external iscntrl
    push ax
    call iscntrl

*****
ISCOLOR -- returns true|false if color or mono screen (CPU needed: 8086)
    on entry: nothing
    on exit: zero flag = (true|false)

.external iscolor
    call iscolor

*****
ISEGA -- returns true|false if ega screen (CPU needed: 8086)

```

```

    on entry: nothing
    on exit:  zero flag = (true|false)

.external isega
    call isega

*****
ISLOWER -- returns true|false if AL is lowercase char (CPU needed: 8086)
    on entry: al = char
    on exit:  zero flag = (true|false)

.external islower
    push ax
    call islower

*****
ISUPPER -- returns true|false if AL is uppercase char (CPU needed: 8086)
    on entry: al = char
    on exit:  zero flag = (true|false)

.external isupper
    push ax
    call isupper

*****
LTRIM -- left trims all spaces from a string (CPU needed: 8086)
    on entry: offset to string (ds = segment to string)
    on exit:  nothing

.external ltrim
    push offset temp_str
    call ltrim

*****
RTRIM -- right trims all spaces from a string (CPU needed: 8086)
    on entry: offset to string (ds = segment to string)
    on exit:  nothing

.external rtrim
    push offset temp_str
    call rtrim

*****
NOSOUND -- turns off the speaker (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external nosound
    call nosound

*****
NUMOFF -- turns off the num lock (CPU needed: 8086)
    on entry: nothing
    on exit:  nothing

.external numoff
    call numoff

```

```

*****
NUMON -- turns on the num lock (CPU needed: 8086)
  on entry: nothing
  on exit:  nothing

.external numon
      call numon

*****
PRTBIN -- Prints a binary representatin of AX (CPU needed: 8086)
  on entry: ax = word
  on exit:  nothing

.external prtbin
      push ax
      call prtbin

*****
PRTCRLF -- Prints a CR LF to stdout (CPU needed: 8086)
  on entry: nothing
  on exit:  nothing

.external prtcrLf
      call prtcrLf

*****
PRTHEX -- Prints a hexadecimal representatin of AX (CPU needed: 80186)
  on entry: ax = word
  on exit:  nothing

.186
.external prthex
      push ax
      call prthex

*****
PRTHEXH -- Prints a hexadecimal representatin of AX (CPU needed: 80186)
  on entry: ax = word
  on exit:  nothing

.186
.external prthexh
      push ax
      call prthexh ; addes the 'h'

*****
PRTHEXN -- Prints a hexadecimal representatin of AL (CPU needed: 80186)
  on entry: ax = word (ah ignored)
  on exit:  nothing

.186
.external prthexn
      push ax          ; ah ignored
      call prthexn

*****

```

PRTHEXN -- Prints a hexadecimal representatin of AL (CPU needed: 80186)
on entry: ax = word (ah ignored)
on exit: nothing

```
.186
.external prthexnh
    push ax          ; ah ignored
    call prthexnh   ; adds the 'h'
```

PRTINT -- Prints an integer representatin of AX (CPU needed: 8086)
on entry: ax = word
on exit: nothing

```
.external prtint
    push ax
    call prtint     ; unsigned
```

PRTOCT -- Prints an octal representatin of AX (CPU needed: 8086)
on entry: ax = word
on exit: nothing

```
.external prtoct
    push ax
    call prtoct
```

PRTSINT -- Prints an integer representatin of AX (CPU needed: 8086)
on entry: ax = word
on exit: nothing

```
.external prtsint
    push ax
    call prtsint   ; signed
```

PRTSTRING -- Prints an asciiz string (CPU needed: 8086)
on entry: offset to string (ds must point to string also)
on exit: nothing

```
.186
.external prtstring
    push offset temp_string
    call prtsint   ; signed
```

PUTCH -- Send a char to stdout (CPU needed: 8086)
on entry: char
on exit: nothing

```
.external putch
    push ax        ; al = char
    call putch
```

RAND -- Returns a random number. (CPU needed: 8086)

```

on entry: seed
on exit:  ax = random number

.external rand
    push cx                ; seed
    call rand              ; randomize it
    ; ax = random number

*****

STRCPY -- Copies a string to to a buffer. (CPU needed: 8086)
    on entry: segment of source string, offset of source string
             segment of target buffer, offset of target buffer
    on exit:  nothing

.external strcpy
    push offset buffer      ; destination buffer offset
    push ds                ; destination buffer segment
    push offset string      ; source string offset
    push ds                ; source string segment
    call strcpy

*****

STRLEN -- Returns the length of a string. (CPU needed: 8086)
    on entry: segment of string, offset of string
    on exit:  ax = length of string including null byte

.external strlen
    push offset string
    push ds
    call strlen
    ; ax = length including null byte

*****

STRLWR -- Converts a string to lower case. (CPU needed: 8086)
    on entry: segment of string, offset of string
    on exit:  nothing

.external strlwr
    push offset string
    push ds
    call strlwr

*****

STRUPR -- Converts a string to upper case. (CPU needed: 8086)
    on entry: segment of string, offset of string
    on exit:  nothing

.external strupr
    push offset string
    push ds
    call strupr

*****

SOUND -- Turns on Speaker with specified freq. (CPU needed: 8086)
    on entry: freq. desired
    on exit:  nothing

```

```
.external sound
    push word freq
    call sound
```

```
*****
```

```
TOLOWER -- Converts char to lower case. (CPU needed: 8086)
    on entry: char
    on exit:  al = lowercase char
```

```
.external tolower
    push ax
    call tolower
```

```
*****
```

```
TOUPPER -- Converts char to upper case. (CPU needed: 8086)
    on entry: char
    on exit:  al = uppercase char
```

```
.external toupper
    push ax
    call toupper
```

Miscellaneous:

Disclaimer: LibMan is distributed as is...

If LibMan destroys your computer, kills you or your family,
or any other thinkable and unthinkable thing, I take no
responsibility. USE AT YOUR OWN RISK...

The software package is distributed as freeware. No fee except
for small copying fees can be charged. There is no fee for any
file produced with this package as long as this file is for
non-commercial use. If you use this package for commercial
use, a small users fee is required.

However, as an incentive to continue the whole NewBasic package,
a donation in US currency would be very helpful.

Bug Report:

Being that this is a beta version, there are bound to be
bugs in it. Please report bugs to:

fys@cybertrails.com
<http://www.cybertrails.com/~fys/newbasic.htm>

Known Bugs/Errors:

I plan to fix/add the following:

Currently: Numbers are not allowed in filenames
If an error occurred, file name is deleted
Resource File must contain two empty lines
at the end of the file
Add subversion area
and a lot more

Latest Version:

You can get the latest version of NBASM and utilities at:
<http://www.cybertrails.com/~fys/newbasic.htm>