EZLib

COLLABORATORS			
	TITLE : EZLib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 17, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 EZLib **1**

1.1 EZLib documentation . . . 1
1.2 DISCLAIMER . . . 2
1.3 New features . . . 2
1.4 What is ez.lib . . . 3
1.5 Print functions print() kprint() . . . 3
1.6 Data conversion atoi() . . . 5
1.7 Data conversion itoa() . . . 5
1.8 Data conversion sprint() . . . 5
1.9 Data conversion qsprint() . . . 6
1.10 Input gets() . . . 7
1.11 String compare . . . 7
1.12 Character functions . . . 8
1.13 String functions strstr() . . . 9
1.14 String functions insert() . . . 9
1.15 String functions strcat() . . . 9
1.16 String functions strchr() . . . 10
1.17 String functions strcpy() . . . 10
1.18 String functions strlen() . . . 10
1.19 String functions strncat() . . . 10
1.20 String functions strncpy() . . . 11
1.21 String functions strrchr() . . . 11

Chapter 1

EZLib

1.1 EZLib documentation

```
                                                    ez.lib

                          Version 1.4  May '94
                            By Joe Siebenmann



                           DISCLAIMER

                          CHARACTER FUNCTIONS

                          New features

                        What is ez.lib
                           isalnum()  isalpha()
                 isascii()  iscntrl()

                          PRINT FUNCTIONS
                           isdigit()  isgraph()
                 islower()  isprint()
         print()        ispunct()  isspace()
         kprint()       isupper()  isxdigit()


          DATA CONVERSION    STRING FUNCTIONS


                        atoi()

                        strcat()

                        itoa()

                        strchr()

                        sprint()

                        strcpy()
```

```
                qsprint()

                strlen()

                strncat()

                strncpy()
                  INPUT
                strrchr()

                strstr()

                gets()

                insert()

                 STRING COMPARE
                 strcmp()
  strncmp()
  stricmp()
```

1.2 DISCLAIMER

```
      ***  See the Disclaimer in EZAsm.doc  ***


  ------ NO LIABILITY FOR CONSEQUENTIAL DAMAGES ------

  IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DAMAGES
  WHATSOEVER ARISING OUT OF THE USE OF OR INABILITY TO USE
  THESE PROGRAMS.
```

1.3 New features

```
o New functions:

    stricmp()
    gets()
    itoa()
    qsprint()

o Function arguments are pushed onto the stack as usual,
  but now D0-D1/A0-A1 are NOT PRESERVED similar to the
  Amiga functions.  Registers shown under the function
  arguments are only to help with use of "*".

o String functions: strcpy(), strcat(), strncpy() and strncat()
  now return with A0 pointing to the NULL byte location.
```

```
  This can speed up many string operations when used.

o sprint() now returns number of characters processed in D0.

o KPrint(), Print() and AtoI() are now kprint(), print(), and atoi().

o search() is now strstr().
```

1.4 What is ez.lib

```
ez.lib is a scanned library consisting of object files which are
individually loaded if there is an external reference to one of them.
Many "c.lib" like functions are included.  All functions are highly
optimized for speed.  ez.lib can also be linked with high level
language compilers for greatly improved speed.
```

1.5 Print functions print() kprint()

```
Usage:  print( FormatString DataStream [...] )

        kprint( FormatString DataStream [...] )
       A0    A1

Performs string formatting identical to C's printf().

print() outputs the results to the current output handle.
kprint() sends its output out the serial port at 9600 baud.
If you connect another computer that's running a terminal program
with a null modem cable, using kprint(), all the output can be captured
and reviewed even if your program is taking over the display, or crashes!
A great debugging tool!  Also useful while running Enforcer.

More information on RawDoFmt() ( ExecBase ), which these use, can
be found in the AutoDocs.


FormatString

  A C-like null terminated format string.
  The following % options are supported:

  %[flags][width.limit][length]type


  flags  - '-' specifies left justification.
  width  - field width.  If fist character is a '0', field is
      padded with leading zeros.
  limit  - maximum number of characters output from a string ( %s only )
  length - data size:  'l' for LONG  ( All variables MUST be LONG )
```

```
  type   - types supported:

     ld - decimal
     lx - hexadecimal
     s - string
     lc - character


DataStream

  Accepts MULTIPLE variable names, registers, constants etc..

  Constant data ( 2, $f4, 'a', etc. ) is pushed as LONG, so use 'l'.
  ( %ld  %lx  %lc )


Examples:

  print( "word %ld = %s\n" 1 "YHWH" )

  kprint( "foo = %08lx  D2 = %08lx\n" foo d2 )



  When using print() without a DataStream argument, you
  can use it like this:

  print( "Hello, World!\n" )



o A 200 byte buffer for print() is allocated on the stack frame
  at _pbuf.  Be careful when using "%s", too large a string, or
  one accidentally unterminated, could overflow the buffer
  and trash your variables/program.

o print() needs the current output handle, and _DOSBase,
  and will get these automatically.  "OutHandle" will contain
  the current output handle.


o EZAsm supports argument strings surrounded by double quotes.
  Strings are automatically NULL terminated.
  The following C character constants are supported:

    \b  backspace
    \f  form feed
    \n  newline
    \r  carriage return
    \t  horizontal tab
    \v  vertical tab
    \\  backslash
    \"  double quote
    \'  single quote
    \nnn  octal character value
    \xnn  hex character value
```

```
  Examples:

    "Hello, World!\n"

    "\x1b[32mEZAsm  1.8\x1b[39m\n"
```

1.6 Data conversion atoi()

```
Usage:  [D0 =] atoi( String  FormatString )
      A0  A1

Converts ASCII digit characters pointed to by String,
according to the format specified by FormatString.
Conversion is stopped when a non-valid digit character is found.
The result is returned in D0.


FormatString

  "%d"  convert (signed) decimal number
  "%x"  convert hexadecimal number
```

1.7 Data conversion itoa()

```
Usage:  itoa( Buffer FormatString Number )
    A0  A1     D0

Converts Number into ASCII digit characters according
to the format specified in FormatString, and places the
result in Buffer.  D0 will contain the length of the string
in Buffer.



FormatString

  "%d"  convert to ASCII decimal number
  "%x"  convert to ASCII hexadecimal number
```

1.8 Data conversion sprint()

```
Usage:  [D0 =] sprint( Buffer FormatString DataStream [...] )
      A0  A1    A2

Performs string formatting identical to C's printf(), but directs
the output to Buffer.  The number of characters placed in Buffer
is returned in D0, excluding the terminating NULL.
```

```
More information on RawDoFmt() ( ExecBase ), which this uses, can
be found in the AutoDocs.


FormatString

  A C-like null terminated format string.
  The following % options are supported:

  %[flags][width.limit][length]type


  flags  - '-' specifies left justification.
  width  - field width.  If fist character is a '0', field is
      padded with leading zeros.
  limit  - maximum number of characters output from a string ( %s only )
  length - data size:  'l' for LONG  ( All variables MUST be LONG )

  type   - types supported:

     ld - decimal
     lx - hexadecimal
     s - string
     lc - character

DataStream

  Accepts MULTIPLE variable names, registers, constants etc..

  Constant data ( 2, $f4, 'a', etc. ) is pushed as LONG, so use 'l'.
  ( %ld  %lx  %lc )


Example:

  sprint( Buf "%s%s\n" &Path &Name )

  ( see Mk.s for more examples )
```

1.9 Data conversion qsprint()

```
                Usage:    qsprint( Buffer FormatString DataStream [...] )
          A0    A1    A2

Performs string formatting similar to
                 sprint()
                 directing
the output to Buffer.  qsprint() is based on atoi() rather than the much
slower RawDoFmt().  In a typical mixed format operation qsprint()
outperforms sprint() by 1,300 to over 2,600 percent!


FormatString:
```

```
  "%ld" - decimal
  "%lx" - hexadecimal
  "%s"  - string
  "%lc" - character


DataStream:

  Accepts MULTIPLE variable names, registers, constants etc..

  Constant data ( 2, $f4, 'a', etc. ) is pushed as LONG, so
  you must use 'l' ( %ld  %lx  %lc ).
```

1.10 Input gets()

```
Usage:  [D0 =] gets( Buffer )
      A0

Reads characters from stdin into Buffer until a newline character
is read.  The newline character is discarded, and the character
string is NULL terminated.  The Buffer address is returned in D0.
Buffer should not exceed 127 bytes.
```

1.11 String compare

```
Usage:    [D0 =]  strcmp( String1 String2 )

          [D0 =] strncmp( String1 String2 Length )

          [D0 =] stricmp( String1 String2 )
          A0      A1      D0


                 [!] str...( String1 String2 ) {
                                .
                                .
                     }


                 [!] str...( String1 String2 ) {
                                .
                                .
                     } else {
                                .
                                .
                     }
```

```
                [!] str...( String1 String2 ) label


Compares strings pointed to by String1 and String2 ( for strncmp(),
at most, Length characters are compared )  The return value is -1
if String1 was less, 1 if String1 was greater,
and 0 if String1 and String2 match exactly.

stricmp() is a case insensitive string compare ( "Cat" = "cat" )
returning 0 for equal, <0 if String1 was less, and >0 if String1
was greater.

As with the C versions, you'll need to use "!" to "flip" the
zero result when testing for strings being equal.  To test for
not equal, omit the "!".
```

1.12 Character functions

```
isalnum() alphabetic or digit character?
isalpha() alphabetic character?
isascii() ASCII character?  ( $0-$7f )
iscntrl() control character?  ( $0-$1f or $7f )
isdigit() digit character?
isgraph() graphics character? ( $21-$7e )
islower() lowercase letter?
isprint() printable character?  ( including space )
ispunct() punctuation character?
isspace() white space character?  ( $20 $09-$0c )
isupper() uppercase letter?
isxdigit()  hexadecimal digit character?


Usage:

      [D0 =] is.....( CharPtr )


         [!] is.....( CharPtr ) {
                        .
                        .
             }


         [!] is.....( CharPtr ) {
                        .
                        .
             } else {
                        .
                        .
             }


         [!] is.....( CharPtr ) label
```

```
      A0

These functions test a character at the address
pointed to by CharPtr.  If the result is true, D0 will be non-zero,
otherwise D0 = 0.  The result also sets the Z flag.
```

1.13 String functions strstr()

```
Usage:  [D0 =] strstr( String1 String2 )
      A0  A1

Searches the string pointed to by String1 for the first
occurrence of the string pointed to by String2.
Returns a pointer to the located string in D0, if successful.
If unsuccessful, D0 = 0.

( Use this function when searching for two or more characters
( strchr() is better for single character searches ))
```

1.14 String functions insert()

```
Usage:  insert( String1 String2 )
      A0    A1

Inserts the string pointed to by String2, excluding the
terminating NULL byte, starting at the address pointed to
by String1.


String1 will typically be the address returned
by a pervious string search.

String1 must be large enough to hold the resulting string.
```

1.15 String functions strcat()

```
Usage:  strcat( String1 String2 )
      A0    A1

Concatenates the string pointed to by String2 to the string
pointed to by String1.  A NULL byte is placed at the end of
the final string.


A0 will contain the address of the NULL byte.
```

1.16 String functions strchr()

```
Usage: [D0 =] strchr( String Char )
       A0    D0

Searches the string pointed to by String for the first
occurrence of the character Char.  If found, a pointer
to the located character is returned in D0, otherwise D0 = 0.

( be sure the actual character value is loaded into Char, not
its address ( use: 'x' ))
```

1.17 String functions strcpy()

```
Usage:  strcpy( String1 String2 )
      A0   A1

Copies the string pointed to by String2 into the string
pointed to by String1, including the terminating NULL byte.


A0 will contain the address of the NULL byte.

Normally strcat() must loop till it reaches the NULL byte.
With strcpy() leaving its location in A0, a second strcpy()
can perform the same operation as strcat() but much faster.

This enables the replacement of the typical combination:

  strcpy()
  strcat()

with the much faster:

  strcpy()
  strcpy()
```

1.18 String functions strlen()

```
Usage: [D0 =] strlen( String )
       A0

Returns the number of characters in the string pointed
to by String, excluding the terminating NULL byte.
```

1.19 String functions strncat()

```
Usage:  strncat( String1 String2 Length )
       A0    A1    D0

Concatenates the string pointed to by String2 to the string
pointed to by String1 until either the NULL byte is reached,
or Length bytes have been concatenated, whichever occurs first.


A0 will contain the address of the NULL byte.
```

1.20 String functions strncpy()

```
                Usage:  strncpy( String1 String2 Length )
       A0    A1    D0

Copies the string pointed to by String2 into the string
pointed to by String1 until either the NULL byte is reached,
or Length bytes have been copied, whichever occurs first.


A0 will contain the address of the NULL byte.

See
                 strcpy()
```

1.21 String functions strrchr()

```
Usage: [D0 =] strrchr( String Char )
       A0 D0

Searches the string pointed to by String for the last occurrence
of the character Char.  If found, its address is returned in D0,
otherwise D0 = 0.

( be sure the actual character value is loaded into Char, not
its address ( use: 'x' ))
```