

Tos-GNU C Library Functions

This reference card describes the available library functions of GNU C on the Atari ST and is currently based on the library from Jwahr R. Bammi at patchlevel 65. For the most part the functions are alphabetically sorted by function name.

Some of the mentioned functions are only available, when MiNT, the TOS multitasking extension from Eric R. Smith is running, or if you have a TT. The earlier are marked with "(MiNT)", the latter with "(TT)".

BIOS Calls via Trap 13	18
Character Classification and Conversion	4
Date and Time	6
Error Numbers	23
File and Directory Information	7
GEMDOS Calls unique to MiNT	16
GEMDOS Calls unique to the TT	16
GEMDOS Calls via Trap 1	13
I/O Control	4
LineA Calls	22
Mathematical Functions	13
Memory Management	4
Miscellaneous	11
Password and Group Handling	10
Process Management	6
Profiling	12
Regular Expressions	7
Setjmp & Signal Stuff	8
Standard I/O	1
String Handling	5
System Variables and Magic Numbers	23
U*IX Compatibility Routines	8
Value Conversion	11
VT52 Escape Sequences	22
XBIOS Calls unique to the TT	21
XBIOS Calls via Trap 14	19

Additionally, this reference card is postcardware. If you find it usefull, send a postcard to this address: Frank Ridderbusch, Sander Str. 17, W-4790 Paderborn, Germany.

Standard I/O

```
#include <stdio.h>
```

The following aliases for some of the below functions are defined.

getc	fgetc
ungetc	fungetc
putc	fputc
getchar()	fgetc(stdin)
ungetchar(c)	fungetc((c),stdin)
putchar(c)	fputc((c),stdout)

```
void clearerr ( FILE *fp );
```

Clear error and EOF status of stream *fp*. (Macro)

```
int fclose ( FILE *fp );
```

Close the stream *fp*.

```
FILE *fdopen ( int fd, const char *mode );
```

Connect a filepointer to the filedescriptor *fd* with mode *mode*.

```
int feof ( FILE *fp );
```

Test EOF status of stream *fp*. (Macro)

```
int ferror ( FILE *fp );
```

Test error status of stream *fp*. (Macro)

```
int fflush ( FILE *fp );
```

Write data from internal buffer of stream *fp* to the file.

```
int fgetc ( FILE *fp );
```

Get a character from stream *fp*.

```
int fgetpos ( FILE *fp, fpos_t *ptr );
```

Put the current position of the filepointer of stream *fp* into the address *ptr*.

```
char *fgets ( char *buf, int max, FILE *fp );
```

Get a string including the NL character from stream *fp* with *max* byte into *buf*.

```
int fileno ( FILE *fp );
```

Get filedescriptor of stream *fp*. (Macro)

```
FILE *fopen ( const char *fname, const char *mode );
```

Open file *fname* with mode *mode* for stream I/O.

```
int fputc ( int c, FILE *fp );
```

Write a character *c* into stream *fp*.

```
int fputs ( const char *buf, FILE *fp );
```

Write the string in *buf* into stream *fp*.

```
size_t fread ( void *buf, size_t sz, size_t no, FILE *fp );
```

Read *no* items of size *sz* from stream *fp* into *buf*.

```
FILE *freopen ( const char *fname, const char *mode, FILE *fp );
```

Reopen *fp* with file *fname* and mode *mode*.

```
int fseek ( FILE *fp, long pos, int whence );
```

Move the the filepointer from stream *fp* to position *pos* relativ to *whence*.

```
int fsetpos ( FILE *fp, fpos_t *ptr );
```

Set the position of the filepointer of stream *fp* to the value at address *ptr*.

```
long ftell ( FILE *fp );
```

Get the current position of the filepointer from stream *fp*.

```
int fungetc ( int c, FILE *fp );
```

Push the character *c* back onto stream *fp*.

```
size_t fwrite ( const void *buf, size_t size, size_t no, FILE *fp );
```

Write *no* items of size *sz* from *buf* into stream *fp*.

```
long getl ( FILE *fp );
```

Read a long value from stream *fp*.

```
char *gets ( char *buf );
```

Get a string from <stdin> into *buf*. The NL character is discarded.

```
short getw ( FILE *fp );
```

Read a short value from stream *fp*.

```
int pclose ( FILE *fp );
```

Close the stream created by **popen()**.

```
void perror ( const char *str );
```

Display error message starting with *str* depending on *errno*.

```
FILE *popen ( const char *command, const char *mode );
```

Open a pipe with *mode* to *command*. (Faked by temporary files)

```
int printf ( const char *fmt, ... );
int fprintf ( FILE *fp, const char *fmt, ... );
int sprintf ( char *buf, const char *fmt, ... );
int vprintf ( const char *fmt, char * );
int vfprintf ( FILE *fp, const char *fmt, char * );
int vsprintf ( char *buf, const char *fmt, char * );
```

Formatted output to either <stdout>, *fp* or *buf* with fixed or variable number of arguments.

The following table lists the identifying letters of the *fmt* string.

scanf	printf	Meaning
d	d,i	Decimal number
o	o	Octal number
x	x,X	Hexadecimal number
u	u	Unsigned decimal n.
b		Unsigned binary int.
i		Any base num.
D	D	Long Decimal number
O	O	Long Octal number
U	U	Unsigned long decimal
X		Long Hexadecimal
ld,lo,lx	ld,li,lo,lx,lX	Long ... number
lb		Unsigned binary long
li		Any base num. long
hd,ho,hx	d,o,x,X	Short ... number
c	c	Character
s	s	String
	p	Pointer
[...]		Str. consist. of char.
e,f,g	f	Fixed-point number
e,f,g	e,E	Floating-point number
le,lf,lg		Double prec. ... n.
E,F,G		Double prec. ... n.
Le,Lf,Lg	Le,LE,Lf	Long Double ... n.
	g,G,Lg,LG	e/E or f - shortest used

```
long putl ( long c, FILE *fp );
    Output the long value c into the stream fp.

int puts ( const char *buf );
    Write the string in buf to <stdout> appending a NL character.

short putw ( short c, FILE *fp );
    Output the short value c into the stream fp.

int remove ( const char *fname );
    Remove file with name fname.

int rename ( const char *old, const char *new );
    Change name of file from old to new.

void rewind ( FILE *fp );
    Reset the filepointer from stream fp to position 0.

int scanf ( const char *fmt, ... );
int fscanf ( FILE *fp, const char *fmt, ... );
int sscanf ( const char *buf, const char *fmt, ... );
    Formatted input from either <stdin>, fp or buf. (For a description of the letters in the fmt string see printf.)

void setbuf ( FILE *fp, char *buf );
    Set buffer of stream fp to buf.

The following functions are defined as macros.

int setvbuf ( FILE *fp, char *buf, int mode, size_t sz );
    Set buffer of stream fp to buf with size sz and mode mode.

FILE *tmpfile ( void );
    Create a temporary file. File is deleted on normal termination.
```

```
char *tmpnam ( char *buf );
    Create a string for a temporary filename.

void _binmode ( int bool );
    Set default open mode to binary (bool = true) or text.
```

I/O Control

```
#include <ioctl.h>

int ioctl ( int fd, int cmd, void *arg );
    Set or inquire i/o dependend parameter.

int stty ( int fd, struct sgtyb *_tty );
int gtty ( int fd, struct sgtyb *_tty );
    Set or inquire i/o parameters for descriptor fd.

#include <fcntl.h>

int fcntl ( int fd, int cmd, void *arg );
    Set or inquire file i/o dependend parameter (MiNT).
```

Memory Management

```
#include <stdlib.h>

void *calloc ( size_t no, size_t sz );
    Allocate and initialize memory for no items of size size.

void *malloc ( size_t sz );
    Allocate sz bytes of memory.

void *realloc ( void *ptr, size_t sz );
    Resize memory pointed to by ptr to size sz.

void free ( void *ptr );
    Free memory obtained by calloc, malloc or realloc.

void *alloca ( size_t sz );
    Allocate sz bytes from the current function stack.

void _malloczero ( int bool );
    Zero allocated memory blocks. Default == false.

void _mallocChunkSize ( size_t Siz );
    Get memory from OS in atleast Siz sized blocks. Default == 4K

#include <memory.h>

void *clalloc ( long no, long sz );
void *mlalloc ( long sz );
void *relalloc ( void *ptr, long sz );
    Apart from the above functions, these three functions are additionally declared in memory.h and take long arguments.
```

Character Classification and Conversion

The following functions return $\neq 0$, if character *c* belongs to the specified character set; 0 otherwise.

```
#include <ctype.h>

function      set
isalnum(c)    Letter or Digit
isalpha(c)    Letter
isascii(c)    ASCII character (Codes: 0..0177)
iscntrl(c)    Control character (Codes: 0..037, 0177)
isdigit(c)    Digit
isgraph(c)    Printable character except space
islower(c)    Lowercase letter
isprint(c)    Printable character
```

ispunct(*c*) Punctuation char. (not letter, digit or control)
isspace(*c*) Space, Tab, CR, NL, VT, FF or BS
isupper(*c*) Uppercase letter
isxdigit(*c*) Hexadecimal digit
iswhite(*c*) same as **isspace**()
isodigit(*c*) Octal digit
iscymf(*c*) Character starting a C identifier
iscym(*c*) Characters continuing a C identifier

Character conversion functions

function Meaning

toupper(*c*) Convert to uppercase letter
_toupper(*c*) Convert to uppercase letter (macro)
tolower(*c*) Convert to lowercase letter
_tolower(*c*) Convert to lowercase letter (macro)
toint(*c*) Convert digit to integer

String Handling

```
#include <string.h>
```

```
int bcmp ( const void *ptr1, const void *ptr2, size_t no );
    Compare no of bytes at position ptr1 with ptr2.

void bcopy ( const void *src, void *dst, size_t len );
    Copy len bytes from position src to dst.

void bzero ( void *dst, size_t len );
    Set len bytes at position dst to 0.

char *index ( const char *str, int charw );
    Same as strchr.

void *memcpy ( void *dst, const void *src, int ucharstop,
    size_t no );
    Copy characters from src to dst until character ucharstop,
    but copy at most no bytes.

void *memchr ( const void *ptr, int ucharwanted,
    size_t no );
    Find character ucharwanted in area at ptr, but compare
    at most no character.

int memcmp ( const void *ptr1, const void *ptr2,
    size_t no );
    Compare no bytes from memory area at ptr1 with area
    at ptr2.

void *memcpy ( void *dst, const void *src, size_t no );
    Copy no of bytes from area at src to dst.

void *memset ( void *ptr, int ucharfill, size_t no );
    Set no of bytes at position ptr to character ucharfill.

char *rindex ( const char *str, int charw );
    Same as strrchr.

char *strcat ( char *dst, const char *src );
    Concatenate the string at src to the string at dst.

char *strchr ( const char *str, int charw );
    Find first occurrence of character charw in string str.

int strcmp ( const char *str1, const char *str2 );
    Compare string str1 with string str1.

char *strcpy ( char *dst, const char *src );
    Copy string at src to dst.

size_t strcspn ( const char *str, const char *rej );
    Find length of initial segment of str consisting entirely
    of characters not from rej.

char *strdup ( const char *str );
    Return a duplicate of string str.

char *strerror ( int errno );
    Map error number errno to descriptive string.
```

```
size_t strlen ( const char *str );
    Return length of string str.

char *strlwr ( char *str );
    Change all character from string str to lowercase.

char *strncat ( char *dst, const char *src, size_t no );
    Concatenate the string at src to the string at dst, but
    copy at most no character.

int strncmp ( const char *str1, const char *str2, size_t no );
    Compare string str1 with string str1, but compare at
    most no character.

char *strncpy ( char *dst, const char *src, size_t no );
    Copy string at src to dst, but at most sl no character.

char *strpbrk ( const char *str, const char *breakat );
    First occurrence of a character from string breakat in
    string str.

char *strrchr ( const char *str, int charw );
    Find last occurrence of character charw in string str.

char *strrev ( char *str );
    Reverse all charcters in string str. First character be-
    comes last character.

size_t strspn ( const char *str, const char *accept );
    Find length of initial segment of str consisting entirely
    of characters from accept.

char *strstr ( const char *str, const char *wanted );
    Find first occurrence of string wanted in string str.

char *strtok ( char *str, const char *delim );
    Retrieve fields from string str, that are delimited by char-
    acters from string delim.
```

Process Management

```
#include <stdlib.h>

void abort ( );
    Immediately abort the running program.

int atexit ( void (*func)() );
    Register a function for execution on termination.

void exit ( int ret );
    Exit current process with return value ret.

int system ( const char *prog );
    Execute command prog passed as a string.

#include <process.h>

int spawnl ( int mode, char *path, ... );
int spawnv ( int mode, char *path, char **argv );
int spawnle ( int mode, char *path, ... );
int spawnve ( int mode, char *path, char **argv,
    char **envp );
    Spawn a new process. (Only P_WAIT is allowed for mode).

int spawnvp ( int mode, char *name, char **argv );
int spawnlp ( int mode, char *name, ... );
    Execute a program on the default system execution path.
    (The current directory is always searched first.)
```

Date and Time

```
#include <time.h>

char *asctime ( const struct tm *t );
    Convert "time" structure to a string.

clock_t clock ( );
    Return process time used so far in units of CLK_TCK
    ticks per second (under TOS, 200 per second).
```

```
char *ctime ( const time_t *t );
    Convert time in seconds since 1970 to a string.
double difftime ( time_t t1, time_t t2 );
    Return difference between two "time_t" types.
struct tm *gmtime ( const time_t *t );
    Get the Greenwich Mean time.
struct tm *localtime ( const time_t *t );
    Convert given time in seconds since 1970 to local time.
time_t mktime ( const struct tm *t );
    Take a time structure and convert it into representation
    "seconds since midnight January 1 1970, GMT".
size_t strftime ( char *s, size_t maxsize,
    const char *format,
    const struct tm *timeptr );
    Print formatted information about a given time.
time_t time ( time_t *t );
    Return time in seconds since midnight January 1 1970,
    GMT.
void tzset ( );
    Set the timezone and dst flag to the local rules (depend-
    ing on environment variable "TZ").
```

File and Directory Information

```
#include <types.h>
#include <stat.h>
int fstat ( int fd, struct stat *info );
    Obtain information about file associated with filedescrip-
    tor fd.
int lstat ( const char *fname, struct stat *info );
    Like stat but return info about symbolic links.
int stat ( const char *fname, struct stat *info );
    Obtain information about file fname.

#include <dirent.h>
int closedir ( DIR *dirp );
    Close directory associated with dirp.
DIR *opendir ( const char *dirname );
    Open directory dirname for reading.
struct dirent *readdir ( DIR *dirp );
    Read an entry from directory associated with dirp.
void rewinddir ( DIR *dirp );
    Seek to the beginning of directory associated with dirp.
void seekdir ( DIR *dirp, off_t loc );
    Seek to position loc in the directory associated with dirp.
off_t telldir ( DIR *dirp );
    Get the current position in directory associated with
    dirp.

#include <ftw.h>
int ftw ( char *path, int (*fn)(), int param);
    Recursively walk tree rooted at path applying function
    fn.
```

Regular Expressions

```
#include <regex.h>
regex_t *regcomp ( char *exp );
    Compile a regular expression into an internal code.
```

```
int regexec ( regexp *prog, char *string, int bolflag );
    Match the previously compiled expression prog against
    the string.
void regsub ( regexp *prog, char *source, char *dest );
    Substitute the previously matched expression prog with
    source. The result is copied to dest.
void regdump ( regexp *r );
    Dump the regular expression r onto <stdout> in vaguely
    comprehensible form.
```

Setjmp & Signal Stuff

```
#include <setjmp.h>
int setjmp ( jmp_buf buf );
    Prepare buffer buf for non local goto via longjmp().
void longjmp ( jmp_buf buf, int rv );
    Return to previously saved context buf with return value
    rv.
int catch ( jmp_buf id, int (*fn)(void));
void throw ( jmp_buf id, int rv );

#include <signal.h>
__Sigfunc signal ( int sig, __Sigfunc func );
    Install the handler func for signal sig.
int raise ( int sig );
long sigsetmask ( long mask );
    (MiNT)
long sigblock ( long mask );
    (MiNT)
```

U*IX Compatibility Routines

```
#include <unistd.h>
void _exit ( int ret );
    Exit the current process without cleaning up. (This is a
    Pterm())
int access ( const char *fname, int mode );
    Check, if file fname is accessible with mode mode.
int chdir ( const char *dname );
    Change the current working directory to dname including
    drive specification.
int chmod ( const char *fname, int mode );
    Change permissions of file fname to mode.
int chown ( const char *fname, int uid, int gid );
    Change owner and group of file fname. (Fake for now)
int close ( int fd );
    Close the file associated with file descriptor fd.
int creat ( const char *fname, unsigned mode );
    Create the file fname with mode mode.
int dup ( int fd );
    Duplicate filedescriptor fd.
int dup2 ( int fd1, int fd2 );
    Duplicate filedescriptor fd1 to fd2.
int exece ( char *path, char **argv, char **envp );
int execv ( char *path, char **argv );
    Execute a new process from path after a fork().
```

```

int fork ( );
int vfork ( );
    Simulate a non multitasking fork. vfork shares data +
    stack with parent.

char *getcwd ( char *buf, int size );
char *getwd ( char *buf );
    Get the current working directory.

int getgid ( );
int getuid ( );
    Get user and group id's. (Fake for now, returning always
    root).

int getegid ( );
int geteuid ( );
    Get effective user and group id's.

char *getlogin ( );
    Return the user's login name. (Password file and envi-
    ronment are searched in order).

int getopt ( int argc, const char **argv, const char *opt );
    Extract the command line options described by opt from
    argv.

int getpgrp ( );
int setpgrp ( );
    Get or create a process group (This basically a fake; both
    return getpid()).

int getpid ( );
    Return a process id for the currently running process
    (from the basepage).

int getppid ( );
    Get parent process id.

char *initstate ( unsigned seed, char *arg_state, int n );
    Initialize state array of generator for random.

int isatty ( int fd );
    Check, if fd is associated with a terminal (screen).

int kill ( int, int );
int link ( const char *old, const char *new );
    Make a new link from new to old. Always fails.

int symlink ( char *old, char *new );
    Make a new symbolic link from new to old. Env. UNIX-
    MODE determines behavior.

int readlink ( char *fname, char *buf, int siz );
    Read the link for fname into buf.

long lseek ( int fd, long pos, int whence );
    Set the current file position to position pos relative to
    whence.

int mkdir ( const char *dname );
    Make a directory with name dname.

char *mktemp ( char *pattern );
    Return a filename for a temporary file, build after pat-
    tern.

int open ( const char *fname, int mode, ... );
    Open the file fname with mode mode.

void psignal ( int signal, const char *prefix );
    Print a error message describing the signal with a user
    comment.

long random ( );
    Return a pseudo random number in the range from 0 to
    231 - 1.

int rmdir ( const char *dname );
    Remove the directory with name dname.

int read ( int fd, void *buf, int cnt );
long read ( int fd, void *buf, long cnt );
    Read cnt bytes from file descriptor fd into buffer buf.

```

```

void *sbrk ( size_t size );
void *lsbrk ( long size );
    Emulation of the U*ix sbrk() system call.

void setlinebuf ( FILE *fp );
    Change the buffering on stream fp from block/unbuffered
    to line buffered.

int setegid ( int gid );
int seteuid ( int uid );
    Set the effective group and user id.

char *setstate ( char *arg_state );
    Switch state of generator for random.

int sleep ( int n );
    Sleep for n number of seconds.

int srandom ( unsigned int x );
    Seed generator for random.

int stime ( long *time );
    Set the current time to time in U*ix format.

long tell ( int fd );
    Get the current file position of the file associated with
    fd.

int times ( struct tms *buf );
    Get process times into buf.

int umask ( int mode );
    Set access mask. (Fake for now)

int unlink ( const char *fname );
    Remove the file with name fname.

int usleep ( long n );
    Sleep for n number of milliseconds.

int utime ( const char *fname, const struct utimbuf *ftime );
    Set the modification time and date of file fname to ftime.

int write ( int fd, const void *buf, int cnt );
long lwrite ( int fd, const void *buf, long cnt );
    Write cnt bytes from buffer buf to file descriptor fd.

int wait ( int *exit_code );
    Wait for child process.

```

Password and Group Handling

```

#include <unixlib.h>
int endpwent ( );
    Close password file.

int setpwent ( );
    Open password file for further operations.

void setpwfile ( char *fname );
    Use the fname instead of /etc/passwd as password file.

```

The following routines already use the above functions.

```

#include <pwd.h>
struct passwd *getpwent ( );
    Return an entry from the password file.

struct passwd *getpwuid ( int uid );
    Return the entry from the password file for uid.

struct passwd *getpwnam ( char *user );
    Return the entry from the password file for user.

```

```

#include <grp.h>
struct group *getgrgid ( int gid );
    Return the entry from the group file for gid. (Mostly
    fake, returning root or user as groupnames)

```

```
struct group *getgrnam ( char *gname );
    Return the entry from the group file for gname. (Mostly
    fake, returning root or user as groupnames)
```

Value Conversion

```
#include <stdlib.h>

double atof ( const char *str );
    Return double value of number presented in str.

int atoi ( const char *str );
    Return int value of number presented in str.

long atol ( const char *str );
    Return long value of number presented in str.

int abs ( int val );
long labs ( long val );
    Return absolute value of val.

long strtol ( const char *nptr, char **endptr, int base );
unsigned long strtoul ( const char *nptr, char **endptr,
    int base );
    Return the value of the number in nptr with base putting
    the endaddress of the numerical string into endptr.

double strtod ( const char *nptr, const char **endptr );
    Return the double value of the number in nptr putting
    the endaddress of the numerical string into endptr.

#include <support.h>

char *_itoa ( int val, char *buf, int radix );
char *_ltoa ( long val, char *buf, int radix );
char *_ultra ( unsigned long val, char *buf, int radix );
    Convert the value val to a string in buf according to
    radix.

#include <locale.h>

struct lconv *localeconv ( );
    Get rules of the current locale

char *setlocale ( int category, const char *name );
    Define locale named in name.
```

Miscellaneous

```
#include <stdlib.h>

void *bsearch ( const void *key, const void *base,
    size_t num, size_t size, int (*cmp)());
    Binary search a field starting at base with num elements
    each size size for key.

div_t div ( int num, int denom );
ldiv_t ldiv ( long num, long denom );
    Perform division and return both quotient and remain-
    der.

char *getenv ( const char *name );
    Get the contents of the environment variable name.

void qsort ( void *base, size_t num, size_t size, int (*cmp)());
    Quick sort an array of num elements with size size start-
    ing at base.

int rand ( );
    Return a pseudo random number in the range from 0 to
    RAND_MAX.

void srand ( unsigned int seed );
    Use seed to initialize the pseudo-random number gener-
    ator.
```

```
#include <support.h>

int console_input_status ( int handle );
    Check, if character is available from standart CON:.
    (This is the same as Cconis)

unsigned int console_read_byte ( int handle );
    Raw input from standart CON: without echoing to
    screen. (This is the same as Crawcin)

void console_write_byte ( int handle, int character );
    Write character c to handle handle. (This is a Fwrite
    with count 1)

void dos2unix ( const char *t_fname, char *u_fname );
    Convert filenames from TOS format to Uix format.

time_t dostime ( time_t );
    Convert Uix time to TOS time and date (lower word =
    time, upper word = date).

char *findfile ( char *fname, char *path, char **ext );
    Locate the file fname along the paths in path with ext
    containing possible extensions.

void fnmapfunc ( fnmapfunc_t u2dos, fnmapfunc_t dos2u );
    Set mapping functions for the unx2dos and dos2unx
    routines.

time_t unixtime ( unsigned tostime, unsigned todate );
    Convert a TOS time/date pair into a Uix time.

void unx2dos ( const char *u_fname, char *t_fname );
    Convert filenames from Uix format to TOS format.

void _set_unixmode ( char *mode );
    Set features of the extended file system.

void _uniquefy ( char *dos );
    Make an unique TOS filename for the extended file sys-
    tem.

#include <sysvars.h>

long get_sysvar ( void *var );
    Get the contents of the system variable var.

void set_sysvar_to_long ( void *var, long val );
    Set the system variable var to the long value val.
```

Profiling

```
#include <support.h>

also see gprof.ttp and -pg option to gcc

void monstartup ( void *lpc, void *hpc );
    High level interface to profil. [lpc,hpc] is the range to
    sample. Necessary memory is dynamically allocated.

void monitor ( void *lpc, void *hpc, void *buf, size_t bsiz,
    unsigned int nfunc );
    Low level interface to profil. buf is a user supplied buffer.
    PC samples over [lpc,hpc] and mcount() call records are
    accumulated in buf.

void moncontrol ( long flag );
    Selectively control profiling in a program.

int profil ( void *buf, size_t bsiz, size_t off, int shift );
    Execution time profil. Pc is examined every 20ms, (PC-
    off)>> shift if in range, will increment short in buf.
```

Mathematical Functions

```
#include <math.h>

double acos ( double );
double acosh ( double );
double asin ( double );
double asinh ( double );
double atan ( double );
double atan2 ( double, double );
double atanh ( double );
double ceil ( double );
double copysign ( double, double );
double cos ( double );
double cosh ( double );
double dabs ( double );
double exp ( double );
double fabs ( double );
double floor ( double );
double fmod ( double, double );
double frexp ( double, int * );
double ldexp ( double, int );
double log ( double );
double log10 ( double );
double modf ( double, double * );
double poly ( int, double *, double );
double pow ( double, double );
double rint ( double );
double sin ( double );
double sinh ( double );
double sqrt ( double );
double tan ( double );
double tanh ( double );
int matherr ( struct exception * );
int pmlcfs ( int, int );
int pmlcnt ( void );
int pmlerr ( int );
int pmllim ( int );
int pmlsfs ( int, int );
```

GEMDOS Calls via Trap 1

```
#include <osbind.h>
```

The following assignment is valid by default:

Device	handle	
CON:	0	Keyboard
CON:	1	Screen
AUX:	2	Serial port
PRN:	3	Parallel port

```
long Cauxin ( );
    Read character from standart AUX:.

short Caxis ( );
    Check, if character is available from standart AUX:.

short Cauxos ( );
    Check, if character can be written to standart AUX:.

void Cauxout ( short c );
    Write character c to standart AUX:.

long Cconin ( );
    Read a character from standart CON:.

short Cconis ( );
    Check, if character is available from standart CON:.

short Cconos ( );
    Check, if character can be written to standart CON:.
```

```
void Cconout ( short c );
    Write character c to standart CON:.

void Cconrs ( char *buf );
    Read edited data from standart CON:. buf[0] contains
    number of characters minus 1 to read.

void Cconws ( const char *str );
    Write string str to standart CON:.

long Cnecin ( );
    Raw input from standart CON: without echoing to
    screen, but with interpretation of CTRL-S, CTRL-Q and
    CTRL-C.

short Cprnos ( );
    Check, if character can be written to standart PRN:.

void Cprnout ( short c );
    Write character c to standart PRN:.

long Ccawcin ( );
    Raw I/O to standart CON: without echoing to
    screen.

long Ccawio ( short data );
    Raw I/O to standart CON:. If data = 0xff read from
    CON:, otherwise output to CON:.

short Dcreate ( const char *dname );
    Create a directory with name dname.

long Ddelete ( const char *dname );
    Remove a directory with name dname.

long Dfree ( DISKINFO *buf, short drv );
    Get available space of drive drv into buf. drv = 0 means
    current drive.

short Dgetdrv ( );
    Get the number of the default drive.

long Dgetpath ( void *buf, short drv );
    Get the current directory of drive drv. drv = 0 means
    current drive.

long Dsetdrv ( short d );
    Set the number of the default drive.

long Dsetpath ( const char *path );
    Set path for current drive to path.

short Fattrib ( const char *fname, short rwflag, short attr );
    Get (rwflag = 0) or set (rwflag = 1) attribute byte of file
    fname.

long Fclose ( short handle );
    Close the file associated with handle.

struct _dta *Fgetdta ( );
    Get the current disk transfer address.

long Fcreate ( const char *fname, short mode );
    Create file fname with mode mode.

long Fdatetime ( DOSTIME *timeptr, short hdl,
    short rwflg );
    Get (rwflg = 0) or set (rwflg = 1) the time structure of
    the file associated by handle hdl.

long Fdelete ( const char *fname );
    Delete file with name fname.

long Fdup ( short handle );
    Return a second handle for a standart handle (0..5).

long Fforce ( short Hstd, short Hnew );
    Replace one of the standart handles (0..5) with Hnew.

long Fopen ( const char *fname, short mode );
    Open file fname with mode mode.

long Fread ( short handle, long cnt, void *buf );
    Read cnt bytes into buf from file associated with handle.

short Frename ( const char *old, const char *new );
    Change name of file from old to new. Also directories
```

long Fseek (*long pos, short handle, short whence*);
Set access pointer to position *pos* relative to *whence*.

void Fsetdta (*struct _dta *ptr*);
Set the current disk transfer address to *ptr*.

long Ffirst (*const char *filespec, short attr*);
Get first directory slot, which matches *filespec* and *attr*.

long Fsnext ();
Get next match of search started by **Ffirst**.

long Fwrite (*short handle, long cnt, const void *buf*);
Write *cnt* bytes from *buf* to file associated with *handle*.

long Malloc (*long size*);
Allocate *size* bytes of memory.

long Mfree (*void *ptr*);
Release block of memory.

long Mshrink (*long ptr, long size*);
Shrink size of memory block obtained by **Malloc**.

long Pexec (*short mode, const char *prog, const char *tail, char **env*);
Execute *prog* with mode *mode*. The following bits in the **a_ldflags** of the program header modify behaviour:

Bit	Meaning
0	Fastload bit (clear only BSS)
1	load into fast ram (TT)
2	satisfy Malloc (s) from fast ram (TT)
3..27	reserved
28..31	TPA size field (TT)

void Pterm (*short rv*);
Terminate current process with return value *rv*.

void Pterm0 ();
Terminate current and return to calling process.

void Ptermres (*long save, short ret*);
Terminate and stay resident, but keep *save* bytes of memory.

long Super (*void *stack*);
Inquire (*stack* = 1), set (*stack* = 0) or return from (*stack* > 0) supervisor mode.

short Sversion ();
Get the version number of GEMDOS.

short Tgetdate ();
Get the GEMDOS internal date. The hardware clock is read in MEGA ST's.
The return value is composed of the following fields:

Bits	Meaning (Range)
0..4	Day (0..31)
5..8	Month (1..12)
9..15	Year (0..119)(+1980)

short Tgettime ();
Get the GEMDOS internal time.
The return value is composed of the following fields:

Bits	Meaning (Range)
0..4	Seconds (0..29) have to be doubled
5..10	Minutes (0..59)
11..15	Hours (0..23)

long Tsetdate (*short date*);
Set the GEMDOS internal date.

long Tsettime (*short time*);
Set the GEMDOS internal time.

GEMDOS Calls unique to the TT

#include <osbind.h>

long Maddalt (*long start, long size*);
Extend the memory, which is managed by **Mxalloc**() and **Pexec**() with *size* bytes starting at address *start*.

void *Mxalloc (*long amount, short flag*);
Allocate *amount* bytes of memory from the OS pool. -1 for *amount* returns the max. size for an allocatable segment. *flag* defines where the memory is allocated.

Value	Meaning
0	alloc mem only in ST ram
1	alloc mem only in TT ram
2	alloc mem in ST ram, if possible
3	alloc mem in TT ram, if possible

GEMDOS Calls unique to MiNT

#include <mintbind.h>

long Dpathconf (*char *name, short which*);
Return information about various limits/capabilities of the file system on which file *name* is located. The information returned depends upon *which* as follows:

Value	Meaning
-1	max. legal value for <i>which</i>
0	int. limit of open files
1	max. # of links to a file
2	max. len. of a full pathname
3	max. len. of an individual file name
4	# of bytes written atomic. to a FIFO

short Fcntl (*short f, void *arg, short cmd*);
Do file control commands on file handle *f* of an open file. *cmd* and *arg* vary upon the file type of *f* and the operation.

long Fgetchar (*short f, short mode*);
Read a character from the file associated with handle *f*. *mode* determines mode (cooked, raw, echo).

long Finstat (*short f*);
Return the number of characters available for reading from file handle *f*. (Useful only for terminals and pipes)

long Foutstat (*short f*);
Return the number of bytes, that may be written to file handle *f* before the writing process has to go to sleep.

short Fpipe (*short *ptr*);
Create a pipe. *ptr*[0] is the read-only end and *ptr*[1] the write-only end file handle of the pipe.

long Fputchar (*short f, long ch, short mode*);
Write a character to the file whose handle is *f*. *mode* is as for **Fgetchar**.

short Fselect (*unsigned short timeout, long *rfd, long *wfd, long *xfs*);
See, which files are ready for reading, writing or have error conditions on them. *timeout* is the number of milliseconds to wait before returning. A bit in one of the *[rwf]fds* bitmaps, which is on, indicates the file descriptor to be tested.

void Pause ();
Suspend the process until it receives a signal that is not being ignored and is not masked.

short Pdomain (*short newdom*);
 If *newdom* is not -1 , sets the process domain of the current process to *newdom*. Always returns previous domain. Domain 0 is the TOS domain and the default; domain 1 is the MiNT domain.

long Pfork ();
 Create a new process, that is the duplicate of the current one, but has its own copy of the address space. The return value is 0 for the child, and the child's pid in the parent.

short Pgetgid ();
short Psetgid (*short id*);
 Get or set the group id of the current process.

short Pgetpid ();
short Pgetppid ();
short Pgetpgrp ();
 Get the current's process' process id, it's parent's process id, or its process group.

short Pgetuid ();
short Psetuid (*short id*);
 Return or set the user id under which the current process is running.

short Pkill (*short pid*, *short sig*);
 Send signal *sig* to process with given *pid*. If *pid*== 0, the signal is sent to all proc's with the same process group.

void Prusage (*long rsp[8]*);
 Get various resource information from the operating system.

Value	Meaning
r[0]	time spent in MiNT kernel
r[1]	time spent in proc's own code
r[2]	tot. kernel time for children
r[3]	tot. user code time for children
r[4]	memory alloc. for this proc.
r[5]-r[7]	reserved

long Psetlimit (*short lim*, *long val*);
 Get/Set a resource limit for a process. The old limit is returned. If *val* is negativ, the limit is unchanged; 0 sets unlimited resource; any other value sets that limit. *lim* is defined as follows:

Value	Meaning
1	max. CPU for proc in milliseconds
2	max. memory allowed for proc.
3	limit of Malloc ()'ed mem. for proc.

short Psetpgrp (*short pid*, *short newgrp*);
 Set the process group of the process with given *pid* to *newgrp*.

long Psigblock (*unsigned long mask*);
 Adds the signal in the 32 bit *mask* to the blocked set. The return value is the set of previously blocked signals. (Usage: **Psigblock**(1 << SIG#))

long Psignal (*short signal*, (void (*)(handler))(longsig);
 Installs a signal handler for the indicated *signal*. *handler* is the address of a function, that will be called, when the signal occurs.

long Psigpending ();
 Return a longword containing the signals, that have been sent to the process, but not yet handled.

long Psigreturn ();
 Prepare to exit from a signal handler. (only needed with **longjmp**())

long Psigsetmask (*unsigned long mask*);
 Replaces the set of blocked signals with *mask*. The old set of blocked signals is returned.

long Pusrval (*long arg*);
 Return the process specific user value for this process.

short Pvfork ();
 Creates a copy of the current process. Both child and parent share the same address space. The return value is 0 for the child, and the child's pid in the parent.

long Pwait ();
 Return the exit status of the children run asynchronously.

long Pwait3 (*short flag*, *long *rusage*);
 Wait for a child and return its exit status. If *rusage* is non zero, *rusage*[0] contains millisec's spend by child in user space and *rusage*[1] contains millisec's spent by child in kernel space.

short Snice (*short delta*);
 Set current processes "niceness". A nice process has a lower priority.

short Syield ();
 Tell MiNT, that is okay to switch processes right now. Always returns 0.

long Sysconf (*short n*);
 Return information about various limits/capabilities of the current version of MiNT. Possible values for *n* are:

Value	Meaning
-1	max. legal value for <i>n</i>
0	max. # of mem. regions per proc.
1	max. len. of Pexec command line
2	max. # of open files per proc.
3	# of suppl. group id's
4	# of procs per user

long Talarm (*long sec*);
 Set an alarm to go off *sec* from now. If *sec*==0 cancel any pending alarm.

Dos_tab ();
Bios_tab ();
Xbios_tab ();
 These functions return a pointer to the GEMDOS, BIOS and XBIOS funxtion tables.

BIOS Calls via Trap 13

#include <osbind.h>
 Available devices for BIOS calls are:

Device	#	
_PRT	0	parallel port
_AUX	1	serial port
_CON	2	console
_MIDI	3	MIDI port
_IKBD	4	keyboard processor
_RAWCON	5	console (no terminal emulation)

long Bconin (*short dev*);
 Read a character from device *dev*. (Allowed devices: 0...3)

void Bconout (*short dev*, *short c*);
 Write character *c* to device *dev*. (Allowed devices: 0...5)

short Bconstat (*short dev*);
 Get the status of the input device *dev*. (Allowed devices: 1...3).

short Bcostat (*short dev*);
 Get the status of the output device *dev*. (Allowed devices: 0...4)

long **Drvmap** ();
Get the bitmap of available drives.

BPB ***Getbpb** (*short dev*);
Get the BIOS parameter block of device *dev*.

void **Getmpb** (*void *ptr*);
The memory parameter block pointed to by *ptr* is filled.

long **Kbshift** (*short mode*);
Get (*mode* = -1) or set the current keyboard status.
The return value is composed of the following fields:

Bit	Meaning
0	Right shift key
1	Left shift key
2	Control key
3	Alternate key
4	Caps lock key
5	r. mouse button (ClrHome)
6	l. mouse button (Insert)
7	Reserved (0)

short **Mediach** (*short dev*);
Check, if disk in device *dev* has changed.

short **Rwabs** (*short rwflag*, *long buf*, *short n*, *short sector*,
short dev);
Read (*rwflag* = 0) or write (*rwflag* = 1) *n* logical sectors from device *dev*.

void (*)(*void*) **Setexc** (*short vecnum*, *long vecptr*);
Get (*vecptr* = -1) or set an M68000 exception vector.

long **Tickcal** ();
Get the number of milliseconds between two calls of **tickcal**.

XBIOS Calls via Trap 14

#include <osbind.h>

void **Bioskeys** ();
Reset to initial keyboard mapping tables.

short **Blitmode** (*short flag*);
Configure the blitter chip.

short **Cursconf** (*short rate*, *short attr*);
Configure the cursor and blinking speed.

void **Dosound** (*void *ptr*);
Output a number of bytes to the soundchip.

short **Floprd** (*void *buf*, *long filler*, *short drv*, *short sect*,
short trk, *short side*, *short n*);
Read physically *n* sectors from floppy drive *drv*.

short **Flopwr** (*void *buf*, *long filler*, *short drv*, *short sect*,
short trk, *short side*, *short n*);
Write physically *n* sectors to floppy drive *drv*.

short **Flopfmt** (*void *buf*, *long filler*, *short drv*, *short spt*,
short trk, *short side*, *short i*, *long m*,
short v);
Format track *trk* on drive *drv* with *spt* sectors per track.

short **Flopver** (*void *buf*, *long filler*, *short drv*, *short spt*,
short trk, *short side*, *short n*);
Read a number of sectors and compare with memory.

short **Getrez** ();
Get the current display resolution (0 = low, 1 = mid, 2 = high).

long **Gettime** ();
Get the time from the hardware clock (lower word = time, upper word = date).

short **Giaccess** (*short data*, *short reg*);
Get or set a register of the GI sound chip. Set bit 8 of *reg* for write operation.

void **Ikbdws** (*short cnt*, *void *ptr*);
Write a sequence of *cnt*+1 bytes to the keyboard processor.

void **Initmous** (*short type*, *void *param*, *void *vptr*);
Initialize the mouse cursor routines.

IOREC ***Iorec** (*short ioDEV*);
Get a pointer to a device specific buffer.

void **Jdisint** (*short vnum*);
Disable interrupt *vnum* of the MFP 68901 chip.

void **Jenabint** (*short vnum*);
Enable interrupt *vnum* of the MFP 68901 chip.

KBDVECS ***Kbdvbase** ();
Get a pointer to the KBDVECS structure.

short **Kbrate** (*short delay*, *short rebrate*);
Get or set keyboard repeat rate.

KEYTAB ***Keytbl** (*void *nrml*, *void *shft*, *void *caps*);
Set the addresses of the keyboard mapping tables (-1 = don't change).

void ***Logbase** ();
Get the start address of the logical display memory.

void **Mfpint** (*short vnum*, *void *vptr*);
Set the interrupt vectors of the MFP 68901 chip.

void **Midiws** (*short cnt*, *void *ptr*);
Write a sequence of *cnt*+1 bytes to the MIDI-port.

void **Offgibit** (*short ormask*);
Clear a bit in "PORT A" register of the GI chip.

void **Ongibit** (*short andmask*);
Set a bit in "PORT A" register of the GI chip.

void ***Physbase** ();
Get the start address of the physical display memory.

void **Protobt** (*void *buf*, *long serial*, *short dsktyp*,
short exec);
Create a boot sector in memory.

void **Prtblk** (*void *pblkptr*);
Special hardcopy routine, not unlike the **Scrdmp**() routine.

long **Random** ();
Get a 24 bit random number.

long **Rsconf** (*short baud*, *short flow*, *short uc*, *short rs*,
short ts, *short sc*);
Set the communication parameters for the serial port. The return value is the current configuration of the four register of the UART.

void **Scrdmp** ();
Dump the current screen to the printer.

short **Setcolor** (*short colornum*, *short color*);
Get (*color* = -1) or set contents of color register.

void **Setpalette** (*void *palptr*);
Set the color palette of the video hardware.

short **Setprt** (*short config*);
Get or set the current printer configuration.

The bits of the return value are assigned as follows:

Bit	Off	On
0	Matrix-	Daisy Wheel printer
1	color	monochrom
2	Atari-	Epson printer
3	Draft-	High quality
4	Centronics-	RS232 interface
5	Endless	Single sheet paper

void Setscreen (*void *lscrn, void *pscrn, short rez*);
Set the logical and physical display memory start address and the current screen resolution.

void Settime (*long time_date*);
Set the hardware clock to *time_date* (lower word = time, upper word = date).

void *Ssbrk (*short size*);
Reserve *size* bytes at the end of the physical memory (Dummy routine in all TOS versions).

long Supexec (*void *funcptr*);
Execute a subroutine in supervisor mode.

void Vsync ();
Wait until the verical blank interrupt has bitten.

void Xbtimer (*short timer, short ctrl, long data, void *vptr*);
Set and start the MFP 68901 timer.

XBIOS Calls unique to the TT

I don't have more information about this calls at the moment. Can someone please enlighten me.

```
#include <osbind.h>

long Bconctl ( short opcode, long operand );

long Bconmap ( short dev );

void EgetPalette ( short start, short count, word *palptr );
  Read parts or the complete TT color palette. Starting at register start, count words are transfered to the word field palptr.

short EgetShift ( );
  Get the current contents of the videoshifter control register.

short EsetBank ( short bank );
  Immediatly activate one of the 16 TT color banks and copy the values to the ST(E) compatible color register.

short EsetColor ( short num, short val );
  Set color register num to the value val. If val == -1 the current value is returned.

short EsetGray ( short mode );
  Switch between color (mode == 0) and gray scale presentation.

void EsetPalette ( short start, short count, word *palptr );
  Change parts or the complete TT color palette. From the word field palptr count color register are set, starting at register start.

void EsetShift ( short mode );
  Set the videoshift control register and return the previous contents.

short EsetSmear ( short mode );
  If mode ≠ 0, the sample & hold/smear mode is switched on.
```

LineA Calls

```
#include <linea.h>

void linea0 ( );
  Initialize lineA.

void linea1 ( );
  Put pixel.

int linea2 ( );
  Get pixel.

void linea3 ( );
  Draw line.

void linea4 ( );
  Horizontal line.

void linea5 ( );
  Draw rectangle.

void linea6 ( );
  Filled polygon.

void linea7 ( BBPB *ptr );
  Bit blit.

void linea8 ( );
  Text blit.

void linea9 ( );
  Show mouse.

void lineaa ( );
  Hide mouse.

void lineab ( );
  Mouse form.

void lineac ( void *ptr );
  Undraw sprite.

void linead ( int x, int y, SFORM *sprite, void *ptr );
  Draw sprite.

void lineae ( );
  Copy raster.

void lineaf ( );
  Seed fill.
```

VT52 Escape Sequences

```
#include <vt52.h>
```

Mnemonic	Sequence	Meaning
C_UP	ESC A	cur. one line up
C_DOWN	ESC B	cur. one line down
C_RIGHT	ESC C	cur. one column right
C_LEFT	ESC D	cur. one column left
CLEAR_HOME	ESC E	clear screen, cur. home
HOME	ESC H	cur. home
SCROLL_UP	ESC I	cur. one line up (scroll)
CLEAR_DOWN	ESC J	clear to end of screen
DEL_EOL	ESC K	clear until end of line
INS_LINE	ESC L	insert line
DEL_LINE	ESC M	delete line
CURS_LOC	ESC Y%c%c	set cur. to position
CHAR_COLOR	ESC b%c	set char. color
BG_COLOR	ESC c%c	set background color
C_ON	ESC e	switch cur. on
C_OFF	ESC f	switch cur. off
C_SAVE	ESC j	store cur. position
C_RESTORE	ESC k	restore cur. position
ERASE_L	ESC l	clear cur. line
DEL_BOL	ESC o	clear until beg. of line
REV_ON	ESC p	inverse on
REV_OFF	ESC q	inverse off
WRAP_ON	ESC v	automatic line wrap on
WRAP_OFF	ESC w	automatic line wrap off

System Variables and Magic Numbers

```
#include <sysvars.h>
```

Name	Address
proc_lives	(unsigned long *) 0x380
etv_timer	(void (**)) 0x400
etv_critic	(void (**)) 0x404
etv_term	(void (**)) 0x408
memvalid	(unsigned long *) 0x420
mencntrl	(unsigned char *) 0x424
resvalid	(unsigned long *) 0x426
resvector	(void (**)) 0x42a
phystop	(unsigned long *) 0x42e
_membot	(unsigned long *) 0x432
_memtop	(unsigned long *) 0x436
memval2	(unsigned long *) 0x43a
flock	(short *) 0x43e
seekrate	(short *) 0x440
_timr_ms	(short *) 0x442
_fverify	(short *) 0x444
_bootdev	(short *) 0x446
palmode	(short *) 0x448
defshiftmd	(unsigned char *) 0x44a
sshiftmd	(short *) 0x44c
_v_bas_ad	(void *) 0x44e
vblsem	short *) 0x452
nvbls	(short *) 0x454
_vblqueue	(void (**)) 0x456
colorptr	(short **) 0x45a
_vbclock	(unsigned long *) 0x462
_frclock	(unsigned long *) 0x466
_hz_200	(unsigned long *) 0x4ba
conterm	*((char *) 0x484
savptr	(long *) 0x4a2
_nflops	(short *) 0x4A6
_sysbase	(long *) 0x4f2
_shell_p	(long *) 0x4f6
pun_ptr	(HDINFO *) 0x516
_p_cookies	(long **) 0x5A0
PROC_LIVES_MAGIC	0x12345678L
MEMVALID_MAGIC	0x752019F3L
MEMVAL2_MAGIC	0x237698AAL
RESVALID_MAGIC	0x31415926L

Error Numbers

```
#include <errno.h>
```

ENOERR	0	no error
E_OK	ENOERR	
EERROR	1	generic error
EDRNRDY	2	drive not ready
EDRVNR	EDRNRDY	
EUKCMD	3	unknown command
EUNCMD	EUKCMD	
ECRC	4	crc error
E_CRC	ECRC	
EBADREQ	5	bad request
ESEEK	6	seek error
E_SEEK	ESEEK	
EUKMEDIA	7	unknown media
EMEDIA	EUKMEDIA	
ESECTOR	8	sector not found

ESECNF	ESECTOR	
EPAPER	9	no paper
EWRITE	10	write fault
EWRITEF	EWRITE	
EREAD	11	read fault
EREADF	EREAD	
EGENERIC	12	general mishap
EROFS	13	write protect
ECHMEDIA	14	media change
E_CHNG	ECHMEDIA	
EUKDEV	15	unknown device
EUNDEV	EUKDEV	
EBADSEC	16	bad sectors
ENBADSF	EBADSEC	
EIDISK	17	insert disk
EOTHER	EIDISK	
EINVAL	32	invalid function number
ENOENT	33	file not found
ESRCH	ENOENT	pid not found
EPATH	34	path not found
EMFILE	35	too many open files
EACCESS	36	access denied
EACCES	36	access denied
EPERM	EACCESS	
EBADF	37	invalid handle
ENOMEM	39	insufficient memory
EFAULT	40	invalid memory block request
ENXIO	46	invalid drive
EXDEV	48	cross device rename
ENMFILES	49	no more files (Fnext)
ENMFIL	ENMFILES	
ERANGE	62	range error
EDOM	63	domain error
EBADARG	64	range error
		/context unknown
EINTERNAL	65	internal error
EINTRN	EINTERNAL	
ENOEXEC	66	invalid program load format
EPLFMT	ENOEXEC	
ESBLOCK	67	set block failed/growth restraints
EGSBF	ESBLOCK	
EEXIST	80	file exists (open)
ENAMETOOLONG	81	file exists (open)
ENOTTY	82	not a tty (ioctl)
EAGAIN	EDRNRDY	try again later
ENOTDIR	EPATH	* preliminary *

Copyright © 1991 Frank Ridderbusch

This multicolumn format was originally designed by Stephen Gildea and modified to fit the Atari ST library functions by Frank Ridderbusch

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.