# MLGMPIDL: OCaml interface for GMP and MPFR libraries (version 1.2.1)

July 9, 2015

# Contents

# Chapter 1

# Module `Introduction`

# Chapter 2

# Module Mpz : GMP multi-precision integers

```
type 'a tt
type m
```

Mutable tag

```
type f
```

Functional (immutable) tag

```
type t = m tt
```

Mutable multi-precision integer

```
val print : Format.formatter -> 'a tt -> unit
val init : unit -> 'a tt
val init2 : int -> 'a tt
val realloc2 : t -> int -> unit
val set : t -> 'a tt -> unit
val set_si : t -> int -> unit
val set_d : t -> float -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_si : int -> 'a tt
val init_set_d : float -> 'a tt
val _init_set_str : string -> int -> 'a tt
val init_set_str : string -> base:int -> t
val get_si : 'a tt -> nativeint
val get_int : 'a tt -> int
val get_d : 'a tt -> float
val get_d_2exp : 'a tt -> float * int
val _get_str : int -> 'a tt -> string
val get_str : base:int -> 'a tt -> string
```

```
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val add : t -> 'a tt -> 'b tt -> unit
val add_ui : t -> 'a tt -> int -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val sub_ui : t -> 'a tt -> int -> unit
val ui_sub : t -> int -> 'a tt -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_si : t -> 'a tt -> int -> unit
val addmul : t -> 'a tt -> 'b tt -> unit
val addmul_ui : t -> 'a tt -> int -> unit
val submul : t -> 'a tt -> 'b tt -> unit
val submul_ui : t -> 'a tt -> int -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val cdiv_q : t -> 'a tt -> 'b tt -> unit
```

The first parameter holds the quotient.

```
val cdiv_r : t -> 'a tt -> 'b tt -> unit
```

The first parameter holds the remainder.

```
val cdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
```

The two first parameters hold resp. the quotient and the remainder).

```
val cdiv_q_ui : t -> 'a tt -> int -> int
```

The first parameter holds the quotient.

```
val cdiv_r_ui : t -> 'a tt -> int -> int
```

The first parameter holds the remainder.

```
val cdiv_qr_ui : t -> t -> 'a tt -> int -> int
```

The two first parameters hold resp. the quotient and the remainder).

```
val cdiv_ui : 'a tt -> int -> int
val cdiv_q_2exp : t -> 'a tt -> int -> unit
```

The first parameter holds the quotient.

```
val cdiv_r_2exp : t -> 'a tt -> int -> unit
```

The first parameter holds the remainder.

```
val fdiv_q : t -> 'a tt -> 'b tt -> unit
val fdiv_r : t -> 'a tt -> 'b tt -> unit
val fdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
val fdiv_q_ui : t -> 'a tt -> int -> int
```

```
val fdiv_r_ui : t -> 'a tt -> int -> int
val fdiv_qr_ui : t -> t -> 'a tt -> int -> int
val fdiv_ui : 'a tt -> int -> int
val fdiv_q_2exp : t -> 'a tt -> int -> unit
val fdiv_r_2exp : t -> 'a tt -> int -> unit
val tdiv_q : t -> 'a tt -> 'b tt -> unit
val tdiv_r : t -> 'a tt -> 'b tt -> unit
val tdiv_qr : t -> t -> 'a tt -> 'b tt -> unit
val tdiv_q_ui : t -> 'a tt -> int -> int
val tdiv_r_ui : t -> 'a tt -> int -> int
val tdiv_qr_ui : t -> t -> 'a tt -> int -> int
val tdiv_ui : 'a tt -> int -> int
val tdiv_q_2exp : t -> 'a tt -> int -> unit
val tdiv_r_2exp : t -> 'a tt -> int -> unit
val gmod : t -> 'a tt -> 'b tt -> unit
val gmod_ui : t -> 'a tt -> int -> int
val divexact : t -> 'a tt -> 'b tt -> unit
val divexact_ui : t -> 'a tt -> int -> unit
val divisible_p : 'a tt -> 'b tt -> bool
val divisible_ui_p : 'a tt -> int -> bool
val divisible_2exp_p : 'a tt -> int -> bool
val congruent_p : 'a tt -> 'b tt -> 'c tt -> bool
val congruent_ui_p : 'a tt -> int -> int -> bool
val congruent_2exp_p : 'a tt -> 'b tt -> int -> bool
val _powm : t -> 'a tt -> 'b tt -> 'c tt -> unit
val _powm_ui : t -> 'a tt -> int -> 'b tt -> unit
val powm : t -> 'a tt -> 'b tt -> modulo:'c tt -> unit
val powm_ui : t -> 'a tt -> int -> modulo:'b tt -> unit
val pow_ui : t -> 'a tt -> int -> unit
val ui_pow_ui : t -> int -> int -> unit
val root : t -> 'a tt -> int -> bool
val sqrt : t -> 'a tt -> unit
val _sqrtrem : t -> t -> 'a tt -> unit
val sqrtrem : t -> remainder:t -> 'a tt -> unit
val perfect_power_p : 'a tt -> bool
val perfect_square_p : 'a tt -> bool
val probab_prime_p : 'a tt -> int -> int
val nextprime : t -> 'a tt -> unit
val gcd : t -> 'a tt -> 'b tt -> unit
val gcd_ui : t option -> 'a tt -> int -> int
val _gcdext : t -> t -> t -> 'a tt -> 'b tt -> unit
val gcdext : gcd:t -> alpha:t -> beta:t -> 'a tt -> 'b tt -> unit
val lcm : t -> 'a tt -> 'b tt -> unit
val lcm_ui : t -> 'a tt -> int -> unit
val invert : t -> 'a tt -> 'b tt -> bool
```

```
val jacobi : 'a tt -> 'b tt -> int
val legendre : 'a tt -> 'b tt -> int
val kronecker : 'a tt -> 'b tt -> int
val kronecker_si : 'a tt -> int -> int
val si_kronecker : int -> 'a tt -> int
val remove : t -> 'a tt -> 'b tt -> int
val fac_ui : t -> int -> unit
val bin_ui : t -> 'a tt -> int -> unit
val bin_uiui : t -> int -> int -> unit
val fib_ui : t -> int -> unit
val fib2_ui : t -> t -> int -> unit
val lucnum_ui : t -> int -> unit
val lucnum2_ui : t -> t -> int -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_d : 'a tt -> float -> int
val cmp_si : 'a tt -> int -> int
val cmpabs : 'a tt -> 'b tt -> int
val cmpabs_d : 'a tt -> float -> int
val cmpabs_ui : 'a tt -> int -> int
val sgn : 'a tt -> int
val gand : t -> 'a tt -> 'b tt -> unit
val ior : t -> 'a tt -> 'b tt -> unit
val xor : t -> 'a tt -> 'b tt -> unit
val com : t -> 'a tt -> unit
val popcount : 'a tt -> int
val hamdist : 'a tt -> 'b tt -> int
val scan0 : 'a tt -> int -> int
val scan1 : 'a tt -> int -> int
val setbit : t -> int -> unit
val clrbit : t -> int -> unit
val tstbit : 'a tt -> int -> bool
val _import :
  t ->
  (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t ->
  int -> int -> unit
val _export :
  'a tt ->
  int -> int -> (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t
val import :
  dest:t ->
  (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t ->
  order:int -> endian:int -> unit
val export :
  'a tt ->
  order:int ->
  endian:int -> (int, Bigarray.int32_elt, Bigarray.c_layout) Bigarray.Array1.t
val fits_int_p : 'a tt -> bool
```

```
val odd_p : 'a tt -> bool
val even_p : 'a tt -> bool
val size : 'a tt -> int
val sizeinbase : 'a tt -> int -> int
val fits_ulong_p : 'a tt -> bool
val fits_slong_p : 'a tt -> bool
val fits_uint_p : 'a tt -> bool
val fits_sint_p : 'a tt -> bool
val fits_ushort_p : 'a tt -> bool
val fits_sshort_p : 'a tt -> bool
```

# Chapter 3

# Module Mpq : GMP multi-precision rationals

```
type 'a tt
type m
```

Mutable tag

```
type f
```

Functional (immutable) tag

```
type t = m tt
```

Mutable multi-precision rationals

```
val canonicalize : 'a tt -> unit
val print : Format.formatter -> 'a tt -> unit
val init : unit -> 'a tt
val set : t -> 'a tt -> unit
val set_z : t -> 'a Mpz.tt -> unit
val set_si : t -> int -> int -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_z : 'a Mpz.tt -> 'b tt
val init_set_si : int -> int -> 'a tt
val init_set_str : string -> base:int -> 'a tt
val init_set_d : float -> 'a tt
val get_d : 'a tt -> float
val set_d : t -> float -> unit
val get_z : Mpz.t -> 'a tt -> unit
val _get_str : int -> 'a tt -> string
val get_str : base:int -> t -> string
val to_string : 'a tt -> string
val to_float : 'a tt -> float
```

```
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val of_frac : int -> int -> 'a tt
val of_mpz : 'a Mpz.tt -> 'b tt
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> 'c tt
val add : t -> 'a tt -> 'b tt -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val div : t -> 'a tt -> 'b tt -> unit
val div_2exp : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val inv : t -> 'a tt -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_si : 'a tt -> int -> int -> int
val sgn : 'a tt -> int
val equal : 'a tt -> 'b tt -> bool
val get_num : Mpz.t -> 'a tt -> unit
val get_den : Mpz.t -> 'a tt -> unit
val set_num : t -> 'a Mpz.tt -> unit
val set_den : t -> 'a Mpz.tt -> unit
```

# Chapter 4

# Module `Mpf` : GMP multi-precision floating-point numbers

```
type 'a tt
type m
```

Mutable tag

```
type f
```

Functional (immutable) tag

```
type t = m tt
```

Mutable multi-precision floating-point numbers

```
val print : Format.formatter -> 'a tt -> unit
val set_default_prec : int -> unit
val get_default_prec : unit -> int
val init : unit -> 'a tt
val init2 : int -> 'a tt
val get_prec : 'a tt -> int
val set_prec : t -> int -> unit
val set_prec_raw : t -> int -> unit
val set : t -> 'a tt -> unit
val set_si : t -> int -> unit
val set_d : t -> float -> unit
val set_z : t -> 'a Mpz.tt -> unit
val set_q : t -> 'a Mpq.tt -> unit
val _set_str : t -> string -> int -> unit
val set_str : t -> string -> base:int -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> 'b tt
val init_set_si : int -> 'a tt
val init_set_d : float -> 'a tt
val _init_set_str : string -> int -> 'a tt
val init_set_str : string -> base:int -> 'a tt
```

```
val get_d : 'a tt -> float
val get_d_2exp : 'a tt -> float * int
val get_si : 'a tt -> nativeint
val get_int : 'a tt -> int
val get_z : Mpz.t -> 'a tt -> unit
val get_q : Mpq.t -> 'a tt -> unit
val _get_str : int -> int -> 'a tt -> string * int
val get_str : base:int -> digits:int -> 'a tt -> string * int
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val of_string : string -> 'a tt
val of_float : float -> 'a tt
val of_int : int -> 'a tt
val of_mpz : 'a Mpz.tt -> 'b tt
val of_mpq : 'a Mpq.tt -> 'b tt
val is_integer : 'a tt -> bool
val add : t -> 'a tt -> 'b tt -> unit
val add_ui : t -> 'a tt -> int -> unit
val sub : t -> 'a tt -> 'b tt -> unit
val ui_sub : t -> int -> 'a tt -> unit
val sub_ui : t -> 'a tt -> int -> unit
val mul : t -> 'a tt -> 'b tt -> unit
val mul_ui : t -> 'a tt -> int -> unit
val mul_2exp : t -> 'a tt -> int -> unit
val div : t -> 'a tt -> 'b tt -> unit
val ui_div : t -> int -> 'a tt -> unit
val div_ui : t -> 'a tt -> int -> unit
val div_2exp : t -> 'a tt -> int -> unit
val sqrt : t -> 'a tt -> unit
val pow_ui : t -> 'a tt -> int -> unit
val neg : t -> 'a tt -> unit
val abs : t -> 'a tt -> unit
val cmp : 'a tt -> 'b tt -> int
val cmp_d : 'a tt -> float -> int
val cmp_si : 'a tt -> int -> int
val sgn : 'a tt -> int
val _equal : 'a tt -> 'b tt -> int -> bool
val equal : 'a tt -> 'a tt -> bits:int -> bool
val reldiff : t -> 'a tt -> 'b tt -> unit
val ceil : t -> 'a tt -> unit
val floor : t -> 'a tt -> unit
val trunc : t -> 'a tt -> unit
val integer_p : 'a tt -> bool
val fits_int_p : 'a tt -> bool
val fits_ulong_p : 'a tt -> bool
```

```
val fits_slong_p : 'a tt -> bool
val fits_uint_p : 'a tt -> bool
val fits_sint_p : 'a tt -> bool
val fits_ushort_p : 'a tt -> bool
val fits_sshort_p : 'a tt -> bool
```

# Chapter 5

# Module `Mpfr` : MPFR multi-precision floating-point numbers

```
type 'a tt
type round =
  | Near
  | Zero
  | Up
  | Down
  | Away
  | Faith
  | NearAway
type m
```

Mutable tag

```
type f
```

Functional (immutable) tag

```
type t = m tt
```

Mutable multi-precision floating-point numbers

```
val print : Format.formatter -> 'a tt -> unit
val print_round : Format.formatter -> round -> unit
val string_of_round : round -> string
val set_default_rounding_mode : round -> unit
val get_default_rounding_mode : unit -> round
val round_prec : t -> round -> int -> int
val get_emin : unit -> int
val get_emax : unit -> int
val set_emin : int -> unit
val set_emax : int -> unit
val check_range : t -> int -> round -> int
val clear_underflow : unit -> unit
```

```
val clear_overflow : unit -> unit
val clear_nanflag : unit -> unit
val clear_inexflag : unit -> unit
val clear_flags : unit -> unit
val underflow_p : unit -> bool
val overflow_p : unit -> bool
val nanflag_p : unit -> bool
val inexflag_p : unit -> bool
val set_default_prec : int -> unit
val get_default_prec : unit -> int
val init : unit -> 'a tt
val init2 : int -> 'a tt
val get_prec : 'a tt -> int
val set_prec : t -> int -> unit
val set_prec_raw : t -> int -> unit
val set : t -> 'a tt -> round -> int
val set_si : t -> int -> round -> int
val set_d : t -> float -> round -> int
val set_z : t -> 'a Mpz.tt -> round -> int
val set_q : t -> 'a Mpq.tt -> round -> int
val _set_str : t -> string -> int -> round -> unit
val set_str : t -> string -> base:int -> round -> unit
val _strtofr : t -> string -> int -> round -> int * int
val strtofr : t -> string -> base:int -> round -> int * int
val set_f : t -> 'a Mpf.tt -> round -> int
val set_si_2exp : t -> int -> int -> round -> int
val set_inf : t -> int -> unit
val set_nan : t -> unit
val swap : t -> t -> unit
val init_set : 'a tt -> round -> int * 'b tt
val init_set_si : int -> round -> int * 'a tt
val init_set_d : float -> round -> int * 'a tt
val init_set_f : 'a Mpf.tt -> round -> int * 'b tt
val init_set_z : 'a Mpz.tt -> round -> int * 'b tt
val init_set_q : 'a Mpq.tt -> round -> int * 'b tt
val _init_set_str : string -> int -> round -> 'a tt
val init_set_str : string -> base:int -> round -> 'a tt
val get_d : 'a tt -> round -> float
val get_d1 : 'a tt -> float
val get_z_exp : Mpz.t -> 'a tt -> int
val get_z : Mpz.t -> 'a tt -> round -> unit
val _get_str : int -> int -> 'a tt -> round -> string * int
val get_str : base:int -> digits:int -> t -> round -> string * int
val to_string : 'a tt -> string
val to_float : ?round:round -> 'a tt -> float
```

```
val to_mpq : 'a tt -> 'b Mpq.tt
val of_string : string -> round -> 'a tt
val of_float : float -> round -> 'a tt
val of_int : int -> round -> 'a tt
val of_frac : int -> int -> round -> 'a tt
val of_mpz : 'a Mpz.tt -> round -> 'b tt
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> round -> 'c tt
val of_mpq : 'a Mpq.tt -> round -> 'b tt
val add : t -> 'a tt -> 'b tt -> round -> int
val add_ui : t -> 'a tt -> int -> round -> int
val add_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val add_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val sub : t -> 'a tt -> 'b tt -> round -> int
val ui_sub : t -> int -> 'a tt -> round -> int
val sub_ui : t -> 'a tt -> int -> round -> int
val sub_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val sub_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val mul : t -> 'a tt -> 'b tt -> round -> int
val mul_ui : t -> 'a tt -> int -> round -> int
val mul_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val mul_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val mul_2ui : t -> 'a tt -> int -> round -> int
val mul_2si : t -> 'a tt -> int -> round -> int
val mul_2exp : t -> 'a tt -> int -> round -> int
val div : t -> 'a tt -> 'b tt -> round -> int
val ui_div : t -> int -> 'a tt -> round -> int
val div_ui : t -> 'a tt -> int -> round -> int
val div_z : t -> 'a tt -> 'a Mpz.tt -> round -> int
val div_q : t -> 'a tt -> 'a Mpq.tt -> round -> int
val div_2ui : t -> 'a tt -> int -> round -> int
val div_2si : t -> 'a tt -> int -> round -> int
val div_2exp : t -> t -> int -> round -> int
val sqrt : t -> 'a tt -> round -> bool
val sqrt_ui : t -> int -> round -> bool
val pow_ui : t -> 'a tt -> int -> round -> bool
val pow_si : t -> 'a tt -> int -> round -> bool
val ui_pow_ui : t -> int -> int -> round -> bool
val ui_pow : t -> int -> 'a tt -> round -> bool
val pow : t -> 'a tt -> 'b tt -> round -> bool
val neg : t -> 'a tt -> round -> int
val abs : t -> 'a tt -> round -> int
val cmp : 'a tt -> 'b tt -> int
val cmp_si : 'a tt -> int -> int
val cmp_si_2exp : 'a tt -> int -> int -> int
val sgn : 'a tt -> int
```

```
val _equal : 'a tt -> 'b tt -> int -> bool
val equal : 'a tt -> 'b tt -> bits:int -> bool
val nan_p : 'a tt -> bool
val inf_p : 'a tt -> bool
val number_p : 'a tt -> bool
val reldiff : t -> 'a tt -> 'b tt -> round -> unit
val log : t -> 'a tt -> round -> int
val log2 : t -> 'a tt -> round -> int
val log10 : t -> 'a tt -> round -> int
val exp : t -> 'a tt -> round -> int
val exp2 : t -> 'a tt -> round -> int
val exp10 : t -> 'a tt -> round -> int
val cos : 'a tt -> 'b tt -> round -> int
val sin : 'a tt -> 'b tt -> round -> int
val tan : 'a tt -> 'b tt -> round -> int
val sec : 'a tt -> 'b tt -> round -> int
val csc : 'a tt -> 'b tt -> round -> int
val cot : 'a tt -> 'b tt -> round -> int
val sin_cos : 'a tt -> 'b tt -> 'c tt -> round -> bool
val acos : t -> 'a tt -> round -> int
val asin : t -> 'a tt -> round -> int
val atan : t -> 'a tt -> round -> int
val atan2 : t -> 'a tt -> 'b tt -> round -> int
val cosh : 'a tt -> 'b tt -> round -> int
val sinh : 'a tt -> 'b tt -> round -> int
val tanh : 'a tt -> 'b tt -> round -> int
val sech : 'a tt -> 'b tt -> round -> int
val csch : 'a tt -> 'b tt -> round -> int
val coth : 'a tt -> 'b tt -> round -> int
val acosh : t -> 'a tt -> round -> int
val asinh : t -> 'a tt -> round -> int
val atanh : t -> 'a tt -> round -> int
val fac_ui : t -> int -> round -> int
val log1p : t -> 'a tt -> round -> int
val expm1 : t -> 'a tt -> round -> int
val eint : t -> 'a tt -> round -> int
val gamma : t -> 'a tt -> round -> int
val lngamma : t -> 'a tt -> round -> int
val zeta : t -> 'a tt -> round -> int
val erf : t -> 'a tt -> round -> int
val erfc : t -> 'a tt -> round -> int
val j0 : t -> 'a tt -> round -> int
val j1 : t -> 'a tt -> round -> int
val jn : t -> int -> 'a tt -> round -> int
val y0 : t -> 'a tt -> round -> int
```

```
val y1 : t -> 'a tt -> round -> int
val yn : t -> int -> 'a tt -> round -> int
val fma : t -> 'a tt -> 'b tt -> 'c tt -> round -> int
val fms : t -> 'a tt -> 'b tt -> 'c tt -> round -> int
val agm : t -> 'a tt -> 'b tt -> round -> int
val hypot : t -> 'a tt -> 'b tt -> round -> int
val const_log2 : t -> round -> int
val const_pi : t -> round -> int
val const_euler : t -> round -> int
val const_catalan : t -> round -> int
val rint : t -> 'a tt -> round -> int
val ceil : t -> 'a tt -> int
val floor : t -> 'a tt -> int
val round : t -> 'a tt -> int
val trunc : t -> 'a tt -> int
val frac : t -> 'a tt -> round -> int
val modf : t -> t -> 'a tt -> round -> int
val fmod : t -> 'a tt -> 'b tt -> round -> int
val remainder : t -> 'a tt -> 'b tt -> round -> int
val integer_p : 'a tt -> bool
val nexttoward : t -> 'a tt -> unit
val nextabove : t -> unit
val nextbelow : t -> unit
val min : t -> 'a tt -> 'b tt -> round -> int
val max : t -> 'a tt -> 'b tt -> round -> int
val get_exp : 'a tt -> int
val set_exp : t -> int -> int
```

# Chapter 6

# Module Gmp_random : GMP random generation functions

```
type state
val init_default : unit -> state
val init_lc_2exp : 'a Mpz.tt -> int -> int -> state
val init_lc_2exp_size : int -> state
val seed : state -> 'a Mpz.tt -> unit
val seed_ui : state -> int -> unit
module Mpz :
  sig
     val urandomb : Mpz.t -> Gmp_random.state -> int -> unit
     val urandomm : Mpz.t -> Gmp_random.state -> 'a Mpz.tt -> unit
     val rrandomb : Mpz.t -> Gmp_random.state -> int -> unit
  end
module Mpf :
  sig
     val urandomb : Mpf.t -> Gmp_random.state -> int -> unit
  end
module Mpfr :
  sig
     val urandomb : Mpfr.t -> Gmp_random.state -> unit
     val urandom : 'a Mpfr.tt -> Gmp_random.state -> Mpfr.round -> unit
  end
```

# Chapter 7

# Module Mpzf : GMP multi-precision integers, functional version

```
type 'a tt = 'a Mpz.tt
type t = Mpz.f tt
```

multi-precision integer

```
val _mpz : t -> Mpz.t
val _mpzf : Mpz.t -> t
val to_mpz : t -> 'a Mpz.tt
val of_mpz : 'a Mpz.tt -> t
```

Safe conversion from and to Mpz.t.

There is no sharing between the argument and the result.

```
val print : Format.formatter -> 'a tt -> unit
val of_string : string -> t
val of_float : float -> t
val of_int : int -> t
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val add : 'a tt -> 'b tt -> t
val add_int : 'a tt -> int -> t
val sub : 'a tt -> 'b tt -> t
val sub_int : 'a tt -> int -> t
val mul : 'a tt -> 'b tt -> t
val mul_int : 'a tt -> int -> t
val cdiv_q : 'a tt -> 'b tt -> t
val cdiv_r : 'a tt -> 'b tt -> t
val cdiv_qr : 'a tt -> 'b tt -> t * t
val fdiv_q : 'a tt -> 'b tt -> t
val fdiv_r : 'a tt -> 'b tt -> t
val fdiv_qr : 'a tt -> 'b tt -> t * t
val tdiv_q : 'a tt -> 'b tt -> t
val tdiv_r : 'a tt -> 'b tt -> t
```

```
val tdiv_qr : 'a tt -> 'b tt -> t * t
val divexact : 'a tt -> 'b tt -> t
val gmod : 'a tt -> 'b tt -> t
val gcd : 'a tt -> 'b tt -> t
val lcm : 'a tt -> 'b tt -> t
val neg : 'a tt -> t
val abs : 'a tt -> t
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val sgn : 'a tt -> int
```

# Chapter 8

# Module Mpqf : GMP multi-precision rationals, functional version

```
type 'a tt = 'a Mpq.tt
type t = Mpq.f tt
```

multi-precision rationals

```
val to_mpq : t -> 'a Mpq.tt
val of_mpq : 'a Mpq.tt -> t
```

Safe conversion from and to Mpq.t.

There is no sharing between the argument and the result.

```
val _mpq : t -> Mpq.t
val _mpqf : Mpq.t -> t
```

Unsafe conversion from and to Mpq.t.

Sharing between the argument and the result.

```
val print : Format.formatter -> 'a tt -> unit
val of_string : string -> t
val of_float : float -> t
val of_int : int -> t
val of_frac : int -> int -> t
val of_mpz : 'a Mpz.tt -> t
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> t
val to_string : 'a tt -> string
val to_float : 'a tt -> float
val to_mpzf2 : 'a tt -> Mpzf.t * Mpzf.t
val add : 'a tt -> 'b tt -> t
val sub : 'a tt -> 'b tt -> t
val mul : 'a tt -> 'b tt -> t
val div : 'a tt -> 'b tt -> t
val neg : 'a tt -> t
val abs : 'a tt -> t
val inv : 'a tt -> t
```

```
val equal : 'a tt -> 'b tt -> bool
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val cmp_frac : 'a tt -> int -> int -> int
val sgn : 'a tt -> int
val get_num : t -> Mpzf.t
val get_den : t -> Mpzf.t
```

# Chapter 9

# Module Mpfrf : MPFR multi-precision floating-point version, functional version

```
type 'a tt = 'a Mpfr.tt
type t = Mpfr.f tt
```

multi-precision floating-point numbers

```
val to_mpfr : t -> 'a Mpfr.tt
val of_mpfr : 'a Mpfr.tt -> t
```

Safe conversion from and to Mpfr.t.

There is no sharing between the argument and the result.

```
val _mpfr : t -> Mpfr.t
val _mpfrf : Mpfr.t -> t
```

Unsafe conversion from and to Mpfr.t.

The argument and the result actually share the same number: be cautious !

```
val print : Format.formatter -> t -> unit
val of_string : string -> Mpfr.round -> t
val of_float : float -> Mpfr.round -> t
val of_int : int -> Mpfr.round -> t
val of_frac : int -> int -> Mpfr.round -> t
val of_mpz : 'a Mpz.tt -> Mpfr.round -> t
val of_mpz2 : 'a Mpz.tt -> 'b Mpz.tt -> Mpfr.round -> t
val of_mpq : 'a Mpq.tt -> Mpfr.round -> t
val to_string : t -> string
val to_float : ?round:Mpfr.round -> t -> float
val to_mpqf : t -> Mpqf.t
val add : 'a tt -> 'b tt -> Mpfr.round -> t
val add_int : 'a tt -> int -> Mpfr.round -> t
val sub : 'a tt -> 'b tt -> Mpfr.round -> t
val sub_int : 'a tt -> int -> Mpfr.round -> t
```

```
val mul : 'a tt -> 'b tt -> Mpfr.round -> t
val mul_ui : 'a tt -> int -> Mpfr.round -> t
val ui_div : int -> 'b tt -> Mpfr.round -> t
val div : 'a tt -> 'b tt -> Mpfr.round -> t
val div_ui : 'a tt -> int -> Mpfr.round -> t
val sqrt : 'a tt -> Mpfr.round -> t
val ui_pow : int -> 'b tt -> Mpfr.round -> t
val pow : 'a tt -> 'b tt -> Mpfr.round -> t
val pow_int : 'a tt -> int -> Mpfr.round -> t
val neg : 'a tt -> Mpfr.round -> t
val abs : 'a tt -> Mpfr.round -> t
val equal : 'a tt -> 'b tt -> bits:int -> bool
val cmp : 'a tt -> 'b tt -> int
val cmp_int : 'a tt -> int -> int
val sgn : 'a tt -> int
val nan_p : 'a tt -> bool
val inf_p : 'a tt -> bool
val number_p : 'a tt -> bool
```

# Index