```
/*
       ringbuf.h
              August    8,2004      Kenji Arai  for H8/3664F
              August   21,2004
*/

/* ----< Define struct >---------------------------------------------- */
typedef struct {
       unsigned long laptim;        // measured lap time
       unsigned char msr_mode;             // measured by what? (SW, Mg, Ir)
       unsigned char id;                   // Identification Number
} one_lap;

typedef struct {
       struct one_lap *buf_top;     // EEPROM buffer top addr
       struct one_lap *head;        // pointer of ring buffer top
       struct one_lap *tail;        // pointer of ring buffer end
       struct one_lap *rd_ptr;      // pointer of reading top
       int            size;
} lap_ptr;

/* ----< Definition >------------------------------------------------- */
//     definition
#define SIZ           sizeof(one_lap)       // size of one laptimer data
#define MAX_NO LAP_DT_SZ/SIZ // max. number of buffer of laptimer data

#define SW_LAP 0
#define IR_LAP  1
#define MG_LAP 2
#define RD_LAP 3
#define LCD_LAP       4

//     EEPROM definition
#define E24LC256      ( 256 * 1024 / 8 )           // Size of EEPROM
#define EEP_TOP              0
#define SYS_DT_SZ     1024                                 // Size of system data area
#define LAP_DT_SZ     (((E24LC256 -SYS_DT_SZ)/SIZ) * SIZ)
                                                                        // Size of Laptime data
area
#define EEP_LAP_TOP EEP_TOP                                // Lap data top addr
             // System data area
#define EEP_SYS_TOP (EEP_TOP + LAP_DT_SZ)  // Sys data top addr
#define EEP_PTR             (EEP_SYS_TOP + 8)
#define EEP_BSTLAP    (EEP_SYS_TOP + 16)           // 4bytes data / Best(Target) lap data
#define EEP_MINLAP    (EEP_SYS_TOP + 24)           // 4bytes data / Minimum lap data
#define EEP_CLKCMP    (EEP_SYS_TOP + 32)           // 1byte  data / Clock OSC compensatiopn data
#define EEP_BZONOF    (EEP_SYS_TOP + 34)           // 1byte  data / Buzzer on/off flag

/* ----< Function Prototype >----------------------------------------- */
void IdleEEP( void );
void init_eep_lap(void);
void ready_sw_data(void);
void ready_ir_data(void);
void ready_mg_data(void);

void req_lapdata_clear(void);
void req_rd_tglap(void);
void req_wr_tglap(void);
```

```
void req_rd_milap(void);
void req_wr_milap(void);
void req_rd_bzfg(void);
void req_wr_bzfg(void);
void req_rd_osccmp(void);

void rd_eep_trg(void);
void rd_eep_osc(void);
void rd_eep_bz(void);

int    ring_init(lap_ptr *);        // pointer initialize for next power on
int    ring_rd_ptr_top(lap_ptr *);  // pointer initialize for data read
int    ring_rd_ptr_end(lap_ptr *);  // pointer initialize for data read
int ring_rd_ptr_nxt(lap_ptr *);     // pointer change to next read
int ring_rd_ptr_nxt_r(lap_ptr *);   // pointer change to next read
int    ring_is_empty(lap_ptr *);    // check empity condition
int ring_is_end(lap_ptr *);         // check read pointer
int    ring_put_data(lap_ptr *, one_lap *);       // write data into the buffer
int    ring_get_data(lap_ptr *, one_lap *);       // read data from the buffer
int    ring_get_capacity(lap_ptr *);      // check how many data in the buffer
int    ring_clear(lap_ptr *);             // clear EEPROM buffer
int ring_ptr_save(lap_ptr *);       // pointer save into EEPROM
int ring_ptr_load(lap_ptr *);       // pointer read from EEPROM
int ring_data_save(lap_ptr *, one_lap *);  // data write action into EEPROM
int ring_data_load(lap_ptr *, one_lap *);  // data read action from EEPROM
```