The 'bupc_atomic*' function family

```c
#include "bupc_atomic.h"

// Atomic read function
type bupc_atomicX_read_RS(shared void *ptr);

// Atomic set function
void bupc_atomicX_set_RS(shared void *ptr, type val);

// Atomic swap function
type bupc_atomicX_swap_RS(shared void *ptr, type val);

// Conditional swap function
type bupc_atomicX_cswap_RS(shared void *ptr, type oldval, type newval);

// Atomic fetchadd function
type bupc_atomicX_fetchadd_RS(shared void *ptr, type op);

// Atomic fetchand function
type bupc_atomicX_fetchand_RS(shared void *ptr, type op);

// Atomic fetchor function
type bupc_atomicX_fetchor_RS(shared void *ptr, type op);

// Atomic fetchxor function
type bupc_atomicX_fetchxor_RS(shared void *ptr, type op);

// Atomic fetchnot function
type bupc_atomicX_fetchnot_RS(shared void *ptr);
```

Where `type` and `X` take on the values of each pair from the following table, and `RS` is either `strict' or 'relaxed'.

<table>
<thead>
<tr>
<th>Type</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint64_t</td>
<td>U64</td>
</tr>
<tr>
<td>int64_t</td>
<td>I64</td>
</tr>
<tr>
<td>uint32_t</td>
<td>U32</td>
</tr>
<tr>
<td>int32_t</td>
<td>I32</td>
</tr>
</tbody>
</table>

This family of functions provides atomic read, write, and read-modify-write of the indicated data types. When these functions are used to access a memory location in a given synchronization phase, atomicity is guaranteed if and only if no other mechanisms are used to access the same memory location in the same synchronization phase. Memory accesses are relaxed or strict as indicated by the function names.

The `swap` functions set the location given by the first argument to the value of the second argument while atomically returning the prior value. The `cswap` (conditional swap) functions atomically set the location given by the first argument to the value `newval` only if the current value is equal to `oldval`, but return the prior value regardless of whether the write was performed. The `fetchadd` functions atomically add the second argument to the location given by the first argument and return the value prior to the addition. Similarly, the `fetchand`, `fetchor` and `fetchxor` functions atomically perform the appropriate bit-wise operation and return the value prior to the operation. The `fetchnot` functions atomically perform a bit-wise negation of the location given by the argument and return the value prior to the negation.