

NAME

__ppc_get_timebase, __ppc_get_timebase_freq – get the current value of the Time Base Register on Power architecture and its frequency.

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/platform/ppc.h>

uint64_t __ppc_get_timebase(void);
uint64_t __ppc_get_timebase_freq(void);
```

DESCRIPTION

__ppc_get_timebase() reads the current value of the Time Base Register and returns its value, while __ppc_get_timebase_freq() returns the frequency in which the Time Base Register is updated.

The Time Base Register is a 64-bit register provided by Power Architecture processors. It stores a monotonically incremented value that is updated at a system-dependent frequency that may be different from the processor frequency.

RETURN VALUE

__ppc_get_timebase() returns a 64-bit unsigned integer that represents the current value of the Time Base Register.

__ppc_get_timebase_freq() returns a 64-bit unsigned integer that represents the frequency at which the Time Base Register is updated.

STANDARDS

GNU.

HISTORY

__ppc_get_timebase()
glibc 2.16.

__ppc_get_timebase_freq()
glibc 2.17.

EXAMPLES

The following program will calculate the time, in microseconds, spent between two calls to __ppc_get_timebase().

Program source

```
#include <inttypes.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/platform/ppc.h>

/* Maximum value of the Time Base Register: 2^60 - 1.
   Source: POWER ISA. */
#define MAX_TB 0xFFFFFFFFFFFFFFFF

int
main(void)
{
    uint64_t tb1, tb2, diff;
    uint64_t freq;

    freq = __ppc_get_timebase_freq();
    printf("Time Base frequency = %\"PRIu64\" Hz\n", freq);
```

```
    tb1 = __ppc_get_timebase();  
    // Do some stuff...  
    tb2 = __ppc_get_timebase();  
    if (tb2 > tb1) {  
        diff = tb2 - tb1;  
    } else {  
        /* Treat Time Base Register overflow. */  
        diff = (MAX_TB - tb2) + tb1;  
    }  
    printf("Elapsed time = %1.2f usecs\n",  
          (double) diff * 1000000 / freq);  
    exit(EXIT_SUCCESS);  
}
```

SEE ALSO**time(2), usleep(3)**