Power–2 Performance Monitor Software Support

Command Interface

Command Level

load_PMkex [–hqus]

Loads/Unloads/Queries state of kernel extension. The extension must be loaded before either the Command or syscall interface is used. Upon loading the counters are disabled and zeroed and the IMR is set to zero.

–h display help information

–q query state of kernel extension (also exits with nonzero value if loaded)

–u unload kernel extension

–s suppress normal messages

mmcr [–cdhs] [–m mmcr] [–i imr] [mmcr [imr]]

Display current MMCR value. If value parameter is given sets the MMCR to the value specified (in hex).

–h display help information

–c Clear the counters after setting the MMCR

–d same as specifying a value of 0x80000000 (disable counters)

–i set imr to specified value

–m set mmcr to specified value

–s suppress normal messages

–l output one line per entry (for post processing tools)

Command Interface

Command Level

rcount [–cdhlm] [–n number] [–t time] [–i interval] [time [interval [number]]]

Display current counter values. If interval is not specified it defaults to 1 second. –h display help information

–l	output one line per sample (for post processing tools)
–x	display counters in hex
–m	display MMCR in decimal
–n	run for # iterations
–t	run for # seconds
–i	sample interval of # seconds (may be decimal)
–c	clear counters between samples
–1	Skip first sample

ccount [–dh]

Clears the counters

–h	display help information
–d	put 0x80000000 in MMCR (disable counters)
–s	suppress normal messages

These commands are found in /usr/lpp/PMkex.

Note: Any command which reads or modifies the counters will temporarily disables the counters to insure that there is no skew across them.

Command Interface

Example Session – Loading The Kernel Extension. Note that

```
bheema:~/r2perf2% load_PMkex

bheema:~/r2perf2% export PATH=$PATH:/usr/lpp/PMkex

bheema:~/r2perf2% load_PMkex

bheema:~/r2perf2% load_PMkex –q

/usr/lpp/PMkex/PMkex is loaded

bheema:~/r2perf2% load_PMkex –qs && echo ”loaded”

loaded
```

Note: In this example and all those that follow, the humn/machine dialog will be represented by the indentation of this example. The system prompt for these examples is ”system name” : ”current directory” %

Command Interface

Example Session – Set the MMCR to disable the counters, reset the counters to zero and then enable the counters. The counters run, measuring system activity for Monitoring The Sleep Function

```
bheema:~/r2perf2% mmcr –dc ; mmcr 0 ; sleep 5 ; rcount

mmcr = 0 –> 80000000

mmcr = 80000000 –> 0

mmcr   =        0

cycles =   f1d4d5a

scu    =     28f8      2546      1b95       70       42

icu    =     c071    7798e0a    794240c      3820      1e5

fpu    =        6         0        12         0        0

fxu    =   1b526b     a9591      209b       7ed        0
```

Command Interface

Example Session – Set the MMCR 0 to clear and start the counters running in the default setting (mmcr==0), The sleep fucntion is invoked to allow the counters to run 5 seconds. After sleep finishes, rcount then prints the values of the counters to stdout in decimal. This is the simplest method of measuring the system activity (since sleep contributes very little and may be in this case – and most cases – ignored).

```
bheema:~/r2perf2% mmcr –cs 0; sleep 5 ; rcount –d

mmcr   =        0

cycles =  253586536

scu    =     10518      9654      7101       116        71

icu    =     49352  125407292  127154628     14403       483

fpu    =         6         0        18         0         0

fxu    =   1793210    694570      8452      2055         0
```

Command Interface

Example Session – run rcount in background with embedded subroutine calls.

```
bheema:~/r2perf2% rcount –ldcs 0.5 –t 2 > output & ; sleep 2
[1] 7882
[1]  + exit 153   rcount –l –d –s 0.5 –t 2 –c > output
bheema:~/r2perf2% try
mmcr   = 80000000
cycles =     1478
scu    =       0       0       0       0       0
icu    =      28     1cc     cbc       0       0
fpu    =       0       0       0       0       0
fxu    =     938     2d0       0       0       0
...
```

Command Interface

Example Session – unload the kernel extension.

```
bheema:~/r2perf2% load_PMkex –u
bheema:~/r2perf2%
```

Syscall Level

Prototypes

void ph_init (unsigned int MMCR, unsigned int IMR)

> Disables and Clears counters. Stores MMCR for later use. Immediately sets IMR.

void ph_start(void)

> Start counters (by moving stored MMCR into register)

void ph_stop(void)

> Stop counters (by moving 0x80000000 into MMCR register)

void ph_clear(void)

> Clear counters

void ph_read (ph_data *area)

> Read counters into area.

void ph_read_clear (ph_data *area)

> Read counters into area. And clears counters.

int ph_mmcr (int MMCR)

> Set MMCR value and return previous MMCR.

int ph_mmcr_read (void)

> Return current MMCR.

int ph_imr (int IMR)

> Set IMR value and return previous IMR.

int ph_imr_read (void)

> Return current IMR.

Syscall Level (continued)

Prototypes

ph_data Structure

```
typedef struct ph_data ph_data;

struct ph_data {
        unsigned int mmcr;
        unsigned int cycles;
        unsigned int scu[5];
        unsigned int icu[5];
        unsigned int fpu[5];
        unsigned int fxu[5];
        unsigned int serrors;
};
```

Example Program

```
/* try.c: Example of using the Performance monitor syscall interface */
#include <stdlib.h>
#include <stdio.h>
#include "PMkex.h"
void printcounts (ph_data data);
void print5 (unsigned int *data);
void main () {
    int i, j, k, a;
    ph_data data;
    ph_init(0,0); /* Initialize Counters & get ready to use default MMCR/IMR */
    for (j = 0; j < 10; j++) {
        for (i = 0; i < 20; i++) {
            ph_start(); /* Start counters (using above MMCR) */
            for (k = 0; k < 20000; k++)
                a = a * i;
            ph_stop(); /* Stop counters */
            usleep(100);
        }
        ph_read(&data); /* Read counters into data structure */
        printcounts(data); /* Print counters out */
        ph_clear(); /* Clear counters for next run (counters are still running) */
    }
}
```

Example Program (continued)

```
void print5 (unsigned int *data) {
    int i;
    for (i = 0; i < 5; i++)
        printf (" %8x", data[i]);
    printf ("\n");
}
void printcounts (ph_data data) {
    printf ("mmcr = %8x\n", data.mmcr);
    printf ("cycles = %8x\n", data.cycles);
    printf ("scu ="); print5 (data.scu);
    printf ("icu ="); print5 (data.icu);
    printf ("fpu ="); print5 (data.fpu);
    printf ("fxu ="); print5 (data.fxu);
    printf ("serror ="); print5 (data.serror);
}
```

Compiling Example Program

The /usr/lpp/PMkex/PMkex.h file contains prototypes for these system calls.

Use the –bI:/usr/lpp/PMkex/PMkex.exp option when linking.

An example command line could be:

```
cc –O try.c –o try –bI:/usr/lpp/PMkex/PMkex.exp
```