Freescale Semiconductor
Application Note

Document Number: AN4341
Rev. 1, 08/2011

Using the Performance Monitor Unit on the e200z760n3 Power Architecture Core

by: Inga Harris
MSG Application Engineering

Contents

1 Introduction

The e200z7 core, built on Power Architecture® technology with a 10 stage pipeline and dual issue, Harvard architecture core implementation running up to 264 MHz/600 DMIPs has integrated DSP and floating point capability. It also has the capability to reduce code footprints by up to 30% for increased code density and reduced memory size requirements by using Variable Length Encoding (VLE). The e200z7 core, which can be found on devices such as the MPC5674F, has a Performance Monitor Unit (PMU) to support execution profiling. This application note describes the PMU components, their function and how to use this feature to improve the software performance by understanding what the software is doing and exactly where the performance bottleneck happen.

2 Performance Monitor

The PMU provides the application developers the ability to count predefined events and processor clocks associated with particular operations such as cache misses and CPU stalls in order to measure the efficiency of their application software. These counters can be used to trigger interrupts or as data to identify bottlenecks and therefore improve and validate the core performance in the application environment. The Performance Monitor and the registers are described in

Chapter 8 of the Core Reference Manual e200z760RM available at www.freescale.com. As with other specific information about the e200z7 core, all the details of the PMU are documented in the e200z760 Reference Manual. The performance monitor interrupt follows the embedded category in the Power ISA interrupt model and is assigned to interrupt vector offset register 35 (IVOR35). It has the lowest priority of all asynchronous interrupts.

2.1 Performance Monitor Registers

Software interaction with the performance monitor APU is achieved through Performance Monitor Registers (PMRs) rather than SPRs. The PMU also has specific assembler instructions to move to and from the performance monitor registers, mtpmr and mfpmr. The performance monitor mark bit in the Machine State Register (MSR[PMM]) controls which programs are monitored. The system software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR[PR] and MSR[PMM] together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the performance monitor registers, the state for which monitoring is enabled, counting is enabled. Figure 1 shows the PMU Programmer Model.

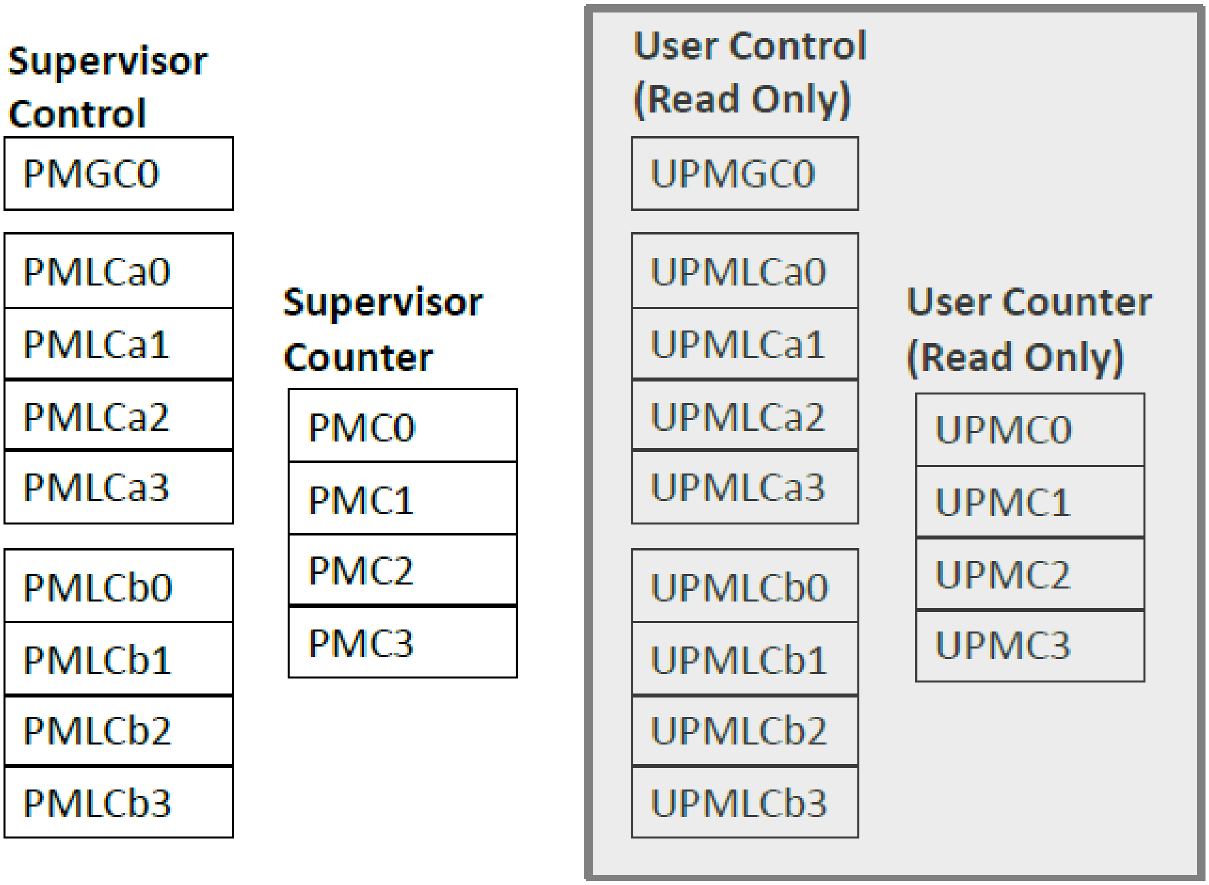


Figure 1. PMU Programmer Model

2.1.1 Control and User Control Registers

The performance monitor global control register, PMGC0, controls the counting of performance monitor events. It takes priority over all the other performance monitor control registers but can only be accessed in supervisor mode. It controls actions such as freezing all counters, enabling interrupts and selecting time bases. UPMGC0 provides user-level read access to PMGC0. The performance monitor local control registers PMLCaN and PMLCbN (where N is 0 to 3) control the individual performance monitor counters, PMCN. Each counter has a corresponding PMLCa and PMLCb register. Again, these can only be accessed in supervisor mode and control the ability to freeze a counter in various modes, enabling overflow conditions and selecting the watchpoint frequency, select trigger events amd thresholds. UPMLCaN and UPMLCbN provide user level read access to PMLCaN and PMLCb0N.

Table 1. Control and Counter Registers

Register Function	Register Name	PMR Number	Register Name[1]	PMR Number
Global Counter Control	PMGC0	400	UPMGC0	384
Local Counter Control a0	PMLCa0	144	UPMLCa0	128
Local Counter Control a1	PMLCa1	145	UPMLCa1	129
Local Counter Control a2	PMLCa2	145	UPMLCa2	130
Local Counter Control a3	PMLCa3	147	UPMLCa3	131
Local Counter Control b0	PMLCb0	272	UPMLCb0	256
Local Counter Control b1	PMLCb1	273	UPMLCb1	257
Local Counter Control b2	PMLCb2	274	UPMLCb2	258
Local Counter Control b3	PMLCb3	275	UPMLCb3	259
Counter 0	PMC0	16	UPMC0	0
Counter 1	PMC1	17	UPMC1	1
Counter 2	PMC2	18	UPMC2	2
Counter 3	PMC3	19	UPMC3	3

[1] User mode registers are Read Only.

2.1.2 Counter Registers

The performance monitor counter registers [PMC0:PMC3] are 32-bit counters used to count software-selectable events. [UPMC0:UPMC3] provide user-level read access to these registers.

3 Writing to the PMRs

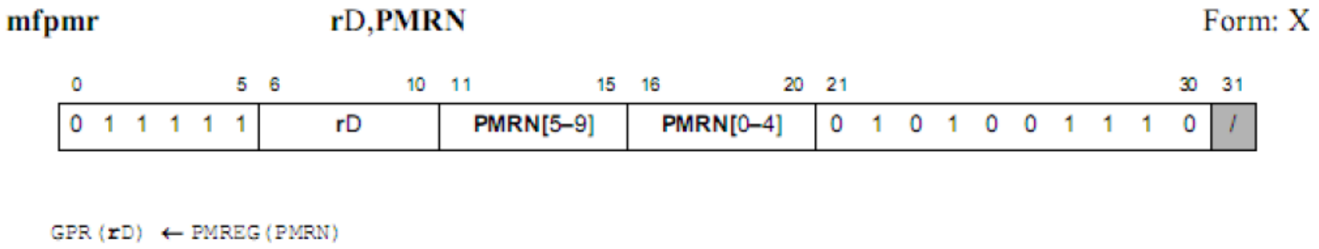
The PMU has two specific assembler instructions to move to and from the performance monitor registers, mfpmr and mtpmr. These two instructions work in the same way as the more common mtspr and mfspr instructions work on the e200z760 Core Special Purpose Registers.

3.1 Move from Performance Monitor Register

The mfpmr instruction moves the content of the performance monitor register designated by PMRN and places them into GPR[rD] as described in Figure below.

Please refer to section 1.3.4 of e200z760RM available at www.freescale.com for an example of how to use this instruction.

Figure 2. mfpmr instruction



mfpmr rD,PMRN Form: X

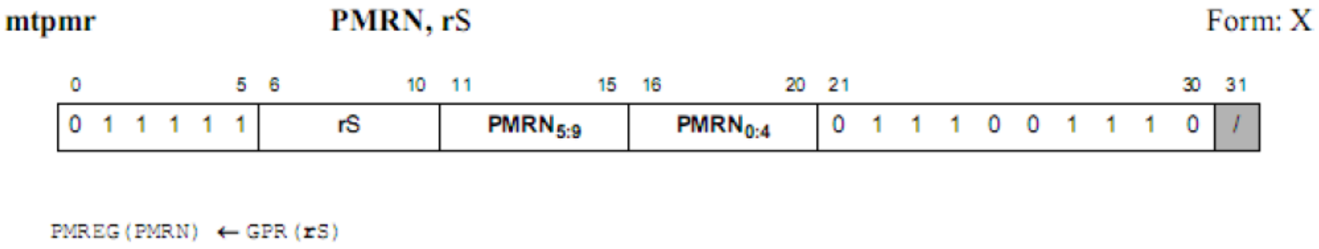
0 5	6 10	11 15	16 20	21 30	31
0 1 1 1 1 1	rD	PMRN[5–9]	PMRN[0–4]	0 1 0 1 0 0 1 1 1 0	/

GPR (rD) ← PMREG (PMRN)

3.2 Move to Performance Monitor Register

The mtpmr instruction moves GPR[rS] in to the performance monitor register designated by PMRN as described in Figure below.

Figure 3. mtpmr instruction



mtpmr PMRN, rS Form: X

0 5	6 10	11 15	16 20	21 30	31
0 1 1 1 1 1	rS	PMRN$_{5:9}$	PMRN$_{0:4}$	0 1 1 1 0 0 1 1 1 0	/

PMREG (PMRN) ← GPR (rS)

4 Example: How to set up the PMU

The following example sets up three Performance Monitor Counters:

- A Pipeline Stall (cycle branch issue stalled) Counter [Com:39]
- A Data Cache Linefills [Com:57]
- An Instruction Cache Linefills [Com:72]
- A Data Cache Linefills [Com:57]
- An Instruction Cache Linefills [Com:72]

The Event Selection Numbers for the events which the PMU can record are listed in Table 8-10 in e200z760RM available from www.freescale.com. An excerpt from the table for the three events above is shown in Table below.

Table 2. Performance Monitor Event Selection

Number	Event	Spec/Non Spec	PR, PMM Filtering	Count Description

Table continues on the next page...

Table 2. Performance Monitor Event Selection (continued)

Pipeline Stalls				
Com: 39	Cycles branch issue stalled	Spec	Yes	Branch held in decode awaiting resolution
Load/Store, Data Cache, and Data Line Fill Events				
Com: 57	Dcache linefills	Spec	Yes	Counts dcache reloads for any reason, including touch-type reloads. Typically used to determine approximate data cache miss rate (along with loads/stores completed).
Fetch, Instruction Cache, Instruction Line Fill, and Instruction Prefetch Events				
Com: 72	Icache linefills	Spec	Yes	Counts icache reloads due to demand fetch. Used to determine instruction cache miss rate (along with instructions completed)

A Performance Monitor Interrupt will generate an IVOR35. This interrupt can be generated by a condition or event if enabled. It is not necessary to have an interrupt for each event being monitored, but an interrupt for a counter overflow is needed for the monitoring of long or multiple tasks. This interrupt routine increments a software overflow counter so that the true number of events can be calculated. Before any application tasks which are to be monitored are started, place the device in to Supervisor Mode, and then disable the counters by freezing the count and clearing the counter. At this point the events can be set up and counting is enabled by unfreezing the counters. Supervisor mode can be disabled if not wanted and the performance tasks can be run. Once the tasks which are being monitored are completed re-enter supervisor mode and freeze the counters. The counters and the overflow counters can be recorded to analyze the devices performance. This process is illustrated in the flow chart in the Figure below.

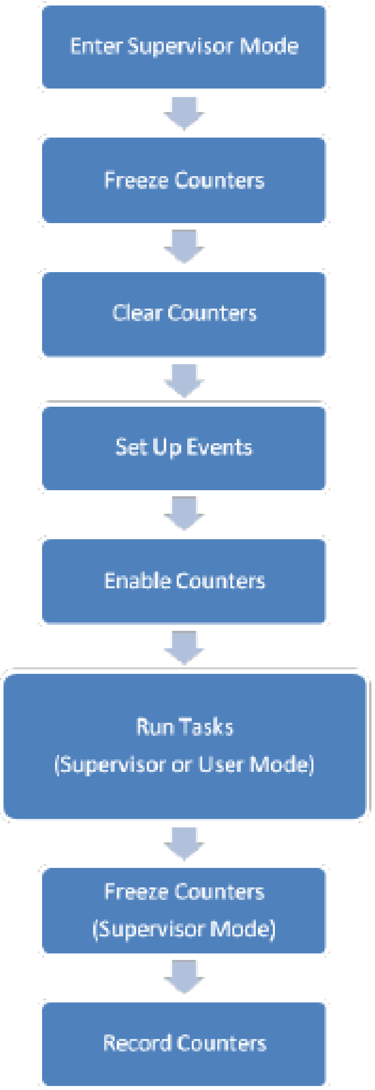


Figure 4. PMU use Flow Chart

4.1 Counter Events

Counter events are listed in the Power Architecture e200 Core family Reference Manual. These are subdivided into the following three groups:

- Reference (Ref:#) - Possible to count these events on any of the four counters (PMC0-PMC3). These events are applicable to most Power Architecture microprocessors
- Common (Com:#) - Possible to count these events on any of the four counters (PMC0-PMC3). These events are specific to the e200 micro-architecture.
- Counter-Specific (C[0-3]:#) - Can only be counted on the specific counter noted. For example, an event assigned to counter PMC2 is shown as C2:#

4.2 Freeze Counters

The counters can be frozen by hardware on an enabled condition or event as described in the e200z760RM available from www.freescale.com, but this example uses the software freeze function. The Global Control Register 0 can be used to freeze all global counters by setting the Freeze All Counters [FAC] bit. The local counters can be frozen individually via the Freeze Counter [FC] bit in their control registers.

```
e_lis 0x8000, r3;
e_or2i 0x0000, r3;
mtpmr400, r3;#freeze all global counters
```

4.3 Enable Counters

Counters will be updated on events if they are not frozen. The freeze bit can be cleared with the following instruction in supervisor mode.

```
e_lis 0, r3;
e_or2i 0, r3;
mtpmr400, r3;#un-freeze all global counters
```

4.4 Set Up Events

The following performance monitors are set up in the local counter and will over flow at 213 events. If this is not enough the interrupt should be enabled and an overflow counter set up so that it is not lost.

```
e_lis 0x0000, r3;
e_or2i 0x2700, r3; #event 39
mtpmr144, r3;#set up pipeline stall event in counter 0
e_lis 0x0000
e_or2i 0x3900, r3; #event 57
mtpmr145, r3;#set up data cache line fills event in counter 1
e_li s0x0000
e_or2i 0x 4800, r3; #event 72
mtpmr146, r3;#set up instr cache line fills event in counter 2
```

This example shows just three events. The events can easily be changed by modifying the value copied in to the PMR. Table 8-10 in the Core Reference Manual describes other possible events such as: Instructions Completed e.g. Event 13, Taken Branch Instructions Completed Pipeline Stalls e.g. Event 40, Cycles execution stalledwaiting for load data BIU Interface Usage e.g. Event 85, BIU instruction-side requests PMC Chaining Events e.g. Event 100, PMC3 rollover Interrupt Events e.g. Event 105, Critical input interrupts taken The Event Selection Number (a value between 0 and 158) given in the table should be placed in bits 8 to 15 of the Control Register.

4.5 Read Counter

The three counters can be read in to general purpose registers (r3, r4 and r5 below) and then collected for analysis. In Supervisor Mode read PMCn:

```
mfpmrr3, 16# read the pipeline stall event counter
mfpmrr4, 17# read the data cache line fills event counter
mfpmrr5, 18# read the instr cache line fills event counter
```

In User Mode read UPMCn:

```
mfpmrr3, 0# read the pipeline stall event counter
mfpmrr4, 1# read the data cache line fills event counter
mfpmrr5, 2# read the instr cache line fills event counter
```

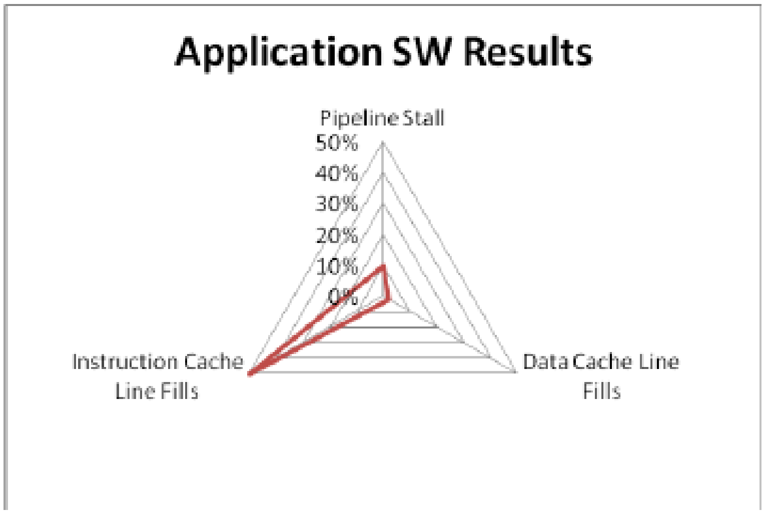
5 How to use the counters to determine efficiency

After integrating the above in the software, the PMR counter values, which are now located in r3, r4 and r5, can be read and stored in variables to be read either in the debugger window or perhaps sent to a terminal via a basic SCI routine. These counter values can then be interpreted. If the above three events were captured over a period of 10 seconds at 200MHz i.e over a 2G CPU clock window and the counters values are:

Counter Percentage	Value
Pipeline Stalls 10%	200,000,000
Data Cache Fills 2%	40,000
Instruction Cache Line Fill 50%	1,000,000,000
Total cycles	2,000,000,000

This implies that the pipleine was stalled on 10% (200M/2G) of instruction loads, the data cache missed on 2% (40M/2G) of CPU clocks and that the instruction cache missed on 50% (1G/2G) of CPU clocks.

Figure 5. Radar Plot of PMU Results



6 Conclusion

The PMU of the Power Architecture® e200z7 core is a powerful feature that enables developers to look inside the core by counting core specific events, for example, cache misses, mispredicted branches, or the number of cycles an execution unit stalls.

7 References

e200z760n3 Power Architecture® Core Reference Manual Rev.0 06/2010 – e200z760RM.pdf available from www.freescale.com

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Document Number: AN4341
Rev. 1, 08/2011